

AFIT/GEE/ENG/99M-01

DEVELOPMENT OF SITE
CHARACTERIZATION SIMULATOR

THESIS

Neil W. Kassel, Captain, USAF

AFIT/GEE/ENG/99M-01

Approved for public release; distribution unlimited.

DTIC QUALITY INSPECTED 4

19990413 123

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 1999	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE Development of Site Characterization Simulator		5. FUNDING NUMBERS		
6. AUTHOR(S) Neil W. Kassel, Captain, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology 2950 P Street WPAFB OH 45433-7765		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GEE/ENG/99M-01		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFIT/CEV 2950 P Street WPAFB OH 45433-7765		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES Mark N. Goltz, PhD				
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <p>The purpose of this study was to develop a tool for Remedial Project Managers (RPMs) to learn and practice the site characterization process. One of the key difficulties encountered by RPMs during site characterization is the need to develop a model of the site based on limited data. These limited data are used to determine parameters that describe hydrogeologic and contaminant properties (aquifer hydraulic conductivity, contaminant concentration, etc.) and are obtained using sampling techniques that interrogate a very small volume, usually much smaller than node spacing of the computer model used to simulate site conditions. The value of the parameter being measured in the field and used in modeling the site is therefore a function of both the sampling location and the volume (of water, aquifer) interrogated. This research identified techniques to incorporate the impact of spatial variability and sampling method (particularly, sampling volume and method error) in a site characterization simulator.</p> <p>This research resulted in the implementation of the Site Characterization Simulator (Sim-Site). Sim-Site "samples" data files, which were created using Groundwater Modeling System (GMS) software, incorporates realistic uncertainty, and delivers the information to the user in a format similar to what a manager would receive during a real investigation.</p>				
14. SUBJECT TERMS site characterization simulator, site characterization, Sim-Site, spatial variability, interpolation, spatial interpolation, kriging, ordinary kriging, MODFLOW, GMS			15. NUMBER OF PAGES 123	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT	

Disclaimer

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

DEVELOPMENT OF SITE CHARACTERIZATION SIMULATOR

THESIS

Neil W. Kassel, Captain, USAF

Presented to the faculty of the Graduate School of Engineering

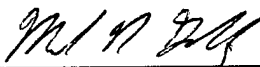
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of


Master of Science in Computer Science



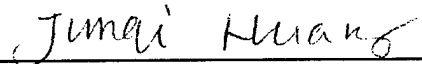
Mark Goltz, Ph.D.
Committee Chairman



Edward Heyse, Maj, USAF, BSC
Committee Member



Thomas Hartrum, Ph.D.
Committee Member



Junqi Huang, Ph.D.
Committee Member

AFIT/GEE/ENG/99M-01

DEVELOPMENT OF SITE CHARACTERIZATION SIMULATOR

THESIS

Presented to the faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Science

Neil W. Kassel, Captain, USAF

March 1999

Approved for public release; distribution unlimited.

Acknowledgments

I would like to thank my thesis advisor, Doctor Mark Goltz, and committee members, Major Ed Heyse, Doctor Junqi Huang, and Doctor Tom Hartrum. Their support and guidance were invaluable during writing of the thesis and the development of the Site Characterization Simulator. A special thanks to Major Heyse for his patience in the early stages of this thesis and for his continued support after he PCSed. Another special thanks to Doctor Goltz for picking up the advisor duties after Major Heyse PCSed.

Most of all, I wish to thank my wife, Tawan, for her patience and encouragement during the many long weekends and evenings that went into the writing of this thesis and developing the Site Characterization Simulator. There are times my wife claimed I was married to my computer because I spent much more time with it than with her.

Neil W. Kassel

Table of Contents

Acknowledgments	ii
List of Figures	viii
List of Tables.....	x
Abstract	xi
1. Introduction	1
1.1 Background	1
1.2 Purpose	1
1.3 Research Questions	2
1.4 Scope of Study	2
1.5 Significance of Study	3
1.6 Overview	3
2. Literature Review	5
2.1 Overview of Spatial Interpolation Techniques.....	5
2.1.1 Knowledge of the Distribution of Natural Resources	6
2.1.2 Using a Single Nearby Value	6
2.1.3 Interpolating from Nearby Values.....	6
2.1.4 Inverse Distance Weighted Averaging (IDWA)	7
2.1.5 Splining	7
2.1.6 Polynomial Regression.....	8
2.1.7 Trend Surface Analysis (TSA).....	8
2.1.8 Kriging	8
2.1.9 Cokriging.....	9
2.2 Comparison of Spatial Interpolation Techniques.....	10
2.2.1 Knowledge of the Distribution of Natural Resources	11
2.2.2 Using a Single Nearby Value	11
2.2.3 Interpolating from Nearby Values.....	11
2.2.4 Inverse Distance Squared (IDA)	11
2.2.5 Optimal Inverse Distance (ODA).....	12
2.2.6 Splining	13
2.2.7 Polynomial Regression.....	13
2.2.8 Trend Surface Analysis (TSA).....	13
2.2.9 Kriging	14
2.2.10 Cokriging.....	16
3. Design Decisions.....	17
3.1 Spatial Interpolation Technique Selection	17
3.1.1 Kriging Technique Selection.....	18
3.1.2 Ordinary Kriging	19
3.2 Programming Language Selection	20

3.2.1	Class Definitions	22
3.2.2	Creating Objects	23
3.2.3	Calling Operations.....	23
3.2.4	Using Inheritance	24
3.2.5	Implementing Associations	25
3.3	GMS Image Files	26
4.	Implementation.....	29
4.1	GMS Files	29
4.1.1	Tabular Scatter Point Files (*.mds).....	29
4.1.2	Map Files (*.map)	30
4.1.3	Image Files (*.img)	31
4.1.4	Other Files	31
4.2	MODFLOW Files	32
4.2.1	Super Files (*.sup).....	32
4.2.2	Output Listing (*.out).....	33
4.2.3	Basic package (*.bas).....	33
4.2.4	Block-Centered Flow Package (*.bcf)	34
4.2.5	Other Files	34
4.3	Kriging Program Translation	35
4.4	Comparison of Visual Basic and FORTRAN Kriging Program	37
4.5	Visual Basic Kriging Program Study	42
5.	Conclusions and Recommendations.....	60
5.1	Kriging	60
5.1.1	Cross Validation.....	60
5.1.2	GMS Variogram Editor	60
5.1.3	Uncertainty Due to Kriging and Estimation Block Size	61
5.2	Sim-Site.....	61
5.2.1	Dimensions.....	61
5.2.2	Sampling Parameters.....	62
5.2.3	Scenario Builder	62
5.2.4	TIFF Image Processing	62
5.2.5	General Enhancements	63
Appendix A: Site Characterization Simulator Specification.....		64
1.	Introduction	64
2.	Objects.....	66
2.1	Start Simulator	68
2.1.1	Properties.....	68
2.1.2	Public Methods.....	68
2.1.3	Private Methods.....	68
2.2	Site Characterization Simulator	68
2.2.1	Properties.....	68
2.2.2	Public Methods.....	68

2.2.3 Private Methods.....	69
2.3 Scenario.....	69
2.3.1 Properties.....	69
2.3.2 Public Methods.....	70
2.3.3 Private Methods.....	71
2.4 Sample.....	71
2.4.1 Properties.....	71
2.4.2 Public Methods.....	71
2.4.3 Private Methods.....	72
2.5 Real Site	72
2.5.1 Properties.....	72
2.5.2 Public Methods.....	72
2.5.3 Private Methods.....	72
2.6 GMS Map.....	72
2.6.1 Properties.....	73
2.6.2 Public Methods.....	73
2.6.3 Private Methods.....	73
2.7 GMS Image	73
2.7.1 Properties.....	73
2.7.2 Public Methods.....	74
2.7.3 Private Methods.....	74
2.8 Grid.....	74
2.8.1 Properties.....	74
2.8.2 Public Methods.....	74
2.8.3 Private Methods.....	74
2.9 Drawing Objects.....	74
2.9.1 Properties.....	75
2.9.2 Public Methods.....	75
2.9.3 Private Methods.....	75
2.10 Well	75
2.10.1 Properties.....	75
2.10.2 Public Methods.....	76
2.10.3 Private Methods.....	76
2.11 Text	76
2.11.1 Properties.....	76
2.11.2 Public Methods.....	76
2.11.3 Private Methods.....	76
2.12 Line.....	77
2.12.1 Properties.....	77
2.12.2 Public Methods.....	77
2.12.3 Private Methods.....	77
2.13 Oval	77
2.13.1 Properties.....	78
2.13.2 Public Methods.....	78
2.13.3 Private Methods.....	78
2.14 Rectangle.....	78

2.14.1 Properties.....	79
2.14.2 Public Methods.....	79
2.14.3 Private Methods.....	79
2.15 Parameter.....	79
2.15.1 Properties.....	80
2.15.2 Public Methods.....	80
2.15.3 Private Methods.....	80
2.16 Method	80
2.16.1 Properties.....	80
2.16.2 Public Methods.....	80
2.16.3 Private Methods.....	81
2.17 User Interface	81
2.17.1 Properties.....	81
2.17.2 Public Methods.....	81
2.17.3 Private Methods.....	81
2.18 Main Window.....	82
2.18.1 Properties.....	82
2.18.2 Public Methods.....	82
2.18.3 Private Methods.....	82
2.18.4 Private Events Used	82
2.19 Results	84
2.19.1 Properties.....	84
2.19.2 Public Methods.....	84
2.19.3 Private Methods.....	84
2.19.4 Private Events Used	85
2.20 User Help.....	85
2.20.1 Properties.....	85
2.20.2 Public Methods.....	85
2.20.3 Private Methods.....	85
2.21 Splash Screen	86
2.21.1 Properties.....	86
2.21.2 Public Methods.....	86
2.21.3 Private Methods.....	86
2.22 Method Choice	86
2.22.1 Properties.....	86
2.22.2 Public Methods.....	86
2.22.3 Private Methods.....	86
2.22.4 Private Events Used	86
2.23 Display Options.....	87
2.23.1 Properties.....	87
2.23.2 Public Methods.....	87
2.23.3 Private Methods.....	87
2.23.4 Private Events Used	87
2.24 About Dialog.....	87
2.25 File.....	88
2.25.1 Properties.....	88

2.25.2 Public Methods.....	88
2.25.3 Private Methods.....	88
2.26 Printer	89
2.26.1 Properties.....	89
2.26.2 Public Methods.....	89
2.26.3 Private Methods.....	89
Appendix B: Coding Conventions	90
1. Introduction	90
2. Coding Conventions Used for Sim-Site	90
2.1 Object Naming Conventions	90
2.2 Constant and Variable Naming Conventions	91
2.3 Structured Coding Conventions	93
Appendix C: Acronyms.....	97
Appendix D: Definitions	99
Bibliography.....	108

List of Figures

Figure 1 Variogram Used in Kriging	20
Figure 3 Kriging Comparison Testing Layout	38
Figure 5 Kriging Comparison Test 2.....	39
Figure 7 Block (3,3) First Selected Points	40
Figure 9 Block (3,3) Possible Choices for Other Four Data Points	40
Figure 11 Data Points Picked by FORTRAN Program.....	42
Figure 13 Data Points Picked by Visual Basic Program.....	42
Figure 15 Sample Gridded Data from GSLIB	43
Figure 17 Grid Locations Used for Simulations	44
Figure 19 Kriging Values Obtained at Location (18.5, 10.5)	45
Figure 21 Execution Times of Different Grid Sizes	46
Figure 23 Kriging Values Obtained at Location (22.0, 12.0)	47
Figure 25 Kriging Values Obtained at Location (14.2, 16.6)	48
Figure 27 The Parameters Used to Define a Model Variogram.....	49
Figure 29 Comparison of Kriging Model Functions.....	50
Figure 16 Execution Time for Kriging Model Functions	51
Figure 32 Effects of Parameter Changes on the Spherical Model	52
Figure 34 Effects of Parameter Changes on the Exponential Model	53
Figure 20 Spherical Model Variogram Fitting.....	55
Figure 21 Gaussian Model Variogram Fitting	55
Figure 22 Exponential Model Variogram Fitting.....	56
Figure 23 Power Model Variogram Fitting.....	56

Figure 24 Comparison of Model Functions Based on Variogram Fitting.....	57
Figure 26 Effects of Decreasing Estimation Block Radius.....	58
Figure 28 Effects of Increasing Estimation Block Radius	59
Figure 30 Sim-Site Object Diagram.....	65

List of Tables

Table 1	Standard GMS File Types Not Currently Used by Sim-Site	31
Table 2	MODFLOW Packages Not Currently Used by Sim-Site.....	34
Table 3	Visual Basic Property Procedures.....	64
Table 4	Prefixes for Visual Basic's Form and Intrinsic Control Objects	90
Table 5	Scope of Visual Basic Variables.....	92
Table 6	Variable Data Type Prefixes	92
Table 7	Information for Procedure Header Comment Blocks	94

Abstract

The purpose of this study was to develop a tool for Remedial Project Managers (RPMs) to learn and practice the site characterization process. One of the key difficulties encountered by RPMs during site characterization is the need to develop a model of the site based on limited data. These limited data are used to determine parameters that describe hydrogeologic and contaminant properties (aquifer hydraulic conductivity, contaminant concentration, etc.) and are obtained using sampling techniques that interrogate a very small volume, usually much smaller than node spacing of the computer model used to simulate site conditions. The value of the parameter being measured in the field and used in modeling the site is therefore a function of both the sampling location and the volume (of water, aquifer) interrogated. This research identified techniques to incorporate the impact of spatial variability and sampling method (particularly, sampling volume and method error) in a site characterization simulator.

This research resulted in the implementation of the Site Characterization Simulator (Sim-Site). Sim-Site "samples" data files, which were created using Groundwater Modeling System (GMS) software, incorporates realistic uncertainty, and delivers the information to the user in a format similar to what a manager would receive during a real investigation.

1. Introduction

1.1 Background

Current methods of characterizing an uncontrolled hazardous waste site consist of estimating certain geological, hydrological, and contaminant parameters throughout the site. The only way to determine these parameters is by analyzing samples from the site. Since budget and time often restrict the number of samples that can be taken, it is important that the right number and kind of samples be taken from the optimal locations, and that the correct properties be analyzed. More importantly, the reason for acquiring these samples is to help the **Remedial Project Manager (RPM)** develop a conceptualization of the system in order to guide decision making. Interpreting these data is perhaps the most difficult part of the site characterization process. Due to the complexity of the subsurface environment and the variety of conditions encountered at different sites, experience may be the best way to gain proficiency at interpreting sample data. A tool for providing site characterization experience, in a safe and economical way, would be quite useful in educating RPMs. A prior study (Heiderscheidt, 1996) provided the basis for developing this tool.

1.2 Purpose

The purpose of this study was to develop a tool for RPMs to learn and practice the site characterization process. Heiderscheidt (1996) identified the geological, hydrological, and contaminant parameters that must be determined to characterize a site. The methods of estimating those parameters (i.e., sampling techniques) and the uncertainty associated with each method was also addressed. In addition, an attempt was made to quantify the **estimation block** and cost of each sampling technique. The

estimation block is the volume of media in the actual site from which the sample is composed, and over which the parameter value, estimated by the sampling method, is averaged.

1.3 *Research Questions*

One of the key difficulties encountered by RPMs during site characterization is the need to develop a model of the site based on limited data. These limited data are obtained using sampling techniques that interrogate a very small volume, usually much smaller than node spacing of the computer model used to simulate site conditions. Therefore, in developing a tool to simulate the site characterization process, a key research question is how to incorporate the impact of spatial variability. That is, at a "real" site, a property value that is obtained by sampling is a function of both the sampling technique (technique error and sample size) and the spatial location of the sample (heterogeneity), since the property varies in space. A simulator must incorporate this variability, so that when a simulator user samples the virtual site for a particular property, the value obtained must also be a function of sample method and sample location.

1.4 *Scope of Study*

This study builds on the results of Heiderscheidt (1996) to develop a site characterization simulator that realistically incorporates the impact of parameter spatial variability and sampling volume. This simulator is intended to be used as a training tool to teach managers how to employ a variety of techniques to get an overall conceptual model of a site, as well as how to translate the data into a complete, quantitative model.

The **Site Characterization Simulator** (Sim-Site) was built from existing **Groundwater Modeling System** (GMS) software. GMS was used to create a "real" site, which was sampled by the user. The user also used GMS to create a model of the site based on sampling data. This project involved writing code to "sample" GMS data files, incorporate realistic uncertainty, and deliver the information to the user in a format similar to what a manager would receive during a real investigation. This project also involved comparing and quantifying how well the user's site model compared to the "real" site.

1.5 Significance of Study

The Civil Engineering and Services School at AFIT (AFIT/CE) is interested in developing a software application that will simulate the site characterization process. Air Force personnel will be able to gain experience in site characterization through training with this software. It will give RPMs the opportunity to practice choosing where to sample, what types of samples to take, and how many samples should be taken, all within a constrained budget. Additionally, it will help personnel learn to use sample data to develop a conceptual model of the site, and ultimately develop a mathematical model. The conceptual and mathematical models are both important tools that assist the RPM in making costly remediation decisions.

1.6 Overview

This thesis consists of four more chapters. Chapter 2 is a review of general literature concerning spatial interpolation that can be applied to Sim-Site. Chapter 3 contains the design decisions used in developing the Sim-Site. Chapter 4 contains the

implementation of the simulator. Finally, Chapter 5 contains results and conclusions about this research, and recommendations for further study.

The appendices of this thesis contain pertinent programming aspects of Sim-Site as well as an elaboration of terms used throughout. Appendix A contains the formal specification of Sim-Site. Appendix B contains the programming conventions used to develop Sim-Site. Appendix C contains a list of acronyms used in this thesis. Finally, Appendix D contains definitions of appropriate geostatistical and computer related terms.

The following typographic conventions are used throughout this thesis:

- The first time a word is encountered that has a definition in Appendix D, it appears in boldface.
- Segments of code are formatted using 10-point Courier New font.
- When an acronym is introduced, the characters that represent the acronym are boldfaced. An example of this is: **Remedial Project Manager** (RPM).
- Words that are italicized are either publications or some type of Visual Basic syntax (i.e., variable, reserved word).

2. Literature Review

In his literature review, Heiderscheidt (1996) introduced the general process of site characterization, and described its inherent difficulties and the research needed regarding uncertainty and variability. His review also covered the role that experience played in reducing uncertainties, and the need for economical methods of obtaining experience.

The purpose of the literature review in this study was to focus more on how to realistically incorporate the impact of parameter spatial variability and sampling volume. When a sample's estimation block is smaller than the node spacing of the real site, Sim-Site is required to report a result to the user. Therefore, a spatial **interpolation** technique is required to estimate the sample parameters.

2.1 Overview of Spatial Interpolation Techniques

Several different techniques exist for spatial interpolation of various types of irregularly scattered data. Most of these interpolation methods create a regular grid of interpolation points from the scattered data. However, Sim-Site is required to return a value from a regular grid, referred to as the "real" site, which was already created by GMS. Therefore, the regular grid now becomes the scattered data. From this, a new grid is created based on the estimation block of the sample method. Interpolation will only be required when the estimation block contains no more than one data point from the "real" site. If the estimation block contains two or more data points, an average of these data points will be used instead of spatial interpolation. To accommodate Sim-Site, the interpolation technique must be able to create a regular grid of any size based on the

estimation block. Sim-Site will then use this grid to obtain a parameter by averaging all data points within the radius of the estimation block.

2.1.1 *Knowledge of the Distribution of Natural Resources*

One technique discussed by Jimenez (1995) requires that we need to know something about how the natural resource was caused. If we can reliably map parent material and geomorphology, we can sample in representative areas of each map unit. We can then use these sample values on the same map unit. In general, this soil survey divides the soil cover into areas that are more homogeneous.

2.1.2 *Using a Single Nearby Value*

One method that Jimenez (1995) discussed was using a single nearby value. If no information is available on the spatial dependence of a variable and primary causes, the most reasonable choice is the nearest measured value. This is similar to **Inverse Distance Weighted Averaging (IDWA)** discussed in Section 2.1.4, which approaches the nearest neighbor interpolation method as the power parameter increases. One way to use this method is to divide the site up into polygons, where each location is associated with its closest observation point.

2.1.3 *Interpolating from Nearby Values*

If variables tend to vary continuously across the site, Jimenez (1995) recommends interpolating a value of an unsampled point from several nearby points. The simplest case considers the three closest points. Another sensible choice uses a distance-weighted average of any number of nearby points, where the closer points receive higher weights. This method is similar to IDWA discussed in Section 2.1.4.

2.1.4 *Inverse Distance Weighted Averaging (IDWA)*

IDWA is a deterministic estimation method where values at unsampled points are determined by a linear combination of values at known sampled points (Collins and Bolstad, 1996). IDWA assumes that data points closer to the location being estimated are more important than samples further away. The weights used depend on the linear distance of the samples from the estimation point. The spatial arrangement of the samples does not affect the weights. IDWA is easy to use and works well with noisy data. The choice of power parameter affects the interpolation results. If the distance of the sample from the estimation point is d , and the power parameter is p , then the resulting weight equation is $(1/d)^p$. Therefore, the power parameter greatly influences the weights used. For instance, IDWA approaches the nearest neighbor interpolation method, where the interpolated value simply takes on the value of the closest sample point, as the power parameter increases. If the power parameter is chosen based on the minimum mean absolute error, then optimal inverse distance **weighting** is achieved. Because of the different variations possible based on the power parameter, this method is evaluated as two separate methods, inverse distance squared and optimal inverse distance, in Section 2.2.

2.1.5 *Splining*

Splining is a deterministic technique to represent two-dimensional curves on three-dimensional surfaces. Splining may be thought of as the mathematical equivalent of fitting a long flexible ruler to a series of data points (Collins and Bolstad, 1996). **Splines** assume smoothness of variation; therefore, they create visually appealing contours. However, splining has no error estimates and may mask data uncertainty.

Splines are typically used to create contour lines from dense regularly-spaced data and may be used for interpolation of irregularly-spaced data.

2.1.6 *Polynomial Regression*

Polynomial regression is a stochastic, global technique that fits the variable of interest to some linear combination of regressor variables (Collins and Bolstad, 1996). The typical goal of this method is to obtain the best fit with the simplest model. The regressor variables have the unwanted effect of increasing multicollinearity, which may negatively affect the model's ability to predict outside the convex hull of data points.

2.1.7 *Trend Surface Analysis (TSA)*

Trend Surface Analysis (TSA), which can be thought of as a subset of polynomial regression, is a stochastic technique that separates the data into regional trends and local variations (Collins and Bolstad, 1996). The regional components are similar to a regression surface fit to the data, while the local variations are similar to a map of residuals. Values may be estimated using a mathematical relationship between locational variables and regionalized variable of interest. TSA differs from polynomial regression in that estimations do not use elevation and use all regressor variables, not a subset. TSA has problems caused by **spatial autocorrelation**, which can yield undesired edge effects and multicollinearity. TSA assumes errors are independent. TSA is useful in removing broad trends prior to further **spatial analysis** such as kriging.

2.1.8 *Kriging*

Kriging is a stochastic technique similar to inverse distance weighted averaging in that it uses a linear combination of weights at known points to estimate the value at an unknown point. Kriging uses a measure of spatial correlation between two points, so the

weights change according to the spatial arrangement of the samples. Unlike the other methods, kriging provides an uncertainty measure of the estimated surface. In addition, kriging does not force a polynomial to fit the data, as TSA does, which results in undesired edge effects. While kriging is considered the **Best Linear Unbiased spatial Predictor (BLUP)**, there are problems of **nonstationarity** in real-world data sets.

Jimenez (1995) points out that the problem with simple interpolation methods is that they can not account for data that are closely spaced in some areas. The idea is that not all data in the same cluster should be used in the distance-weighted formula, because their information is superfluous to some degree. This concept can be formalized and solved by the use of **Best Linear Unbiased Estimator (BLUE)** methods, or what Collins and Bolstad (1996) referred to as BLUP, more commonly known as kriging. These methods provide optimal estimates for each point and the error of the estimate. Therefore, kriging produces the estimated variable and its variance, which can be used directly in error-propagation **Graphical Information Systems (GISs)**. Jimenez (1995) points out that each grid produced by kriging needs a variogram which must be estimated from sample data. However, there is no theoretical basis for a best variogram, so its construction depends greatly on the analyst's skills.

2.1.9 Cokriging

Cokriging is similar to kriging except it uses additional covariates, usually more intensely sampled, to assist in prediction. Cokriging works best when the correlation of the covariates is high.

2.2 *Comparison of Spatial Interpolation Techniques*

Most of the spatial interpolation methods investigated apply to irregularly scattered data and can create a regular grid of interpolated points; however, they all differ in the interpolation schemes they use. Consequently, several comparative analyses of spatial interpolation techniques have been conducted, some of which are relevant to the site characterization process. More specifically, comparative analyses by Collins and Bolstad (1996), Jimenez (1995), and Journel and Huijbregts (1978) were deemed relevant and are therefore summarized in this section.

Collins and Bolstad (1996) performed a comparison of different spatial interpolation techniques in temperature estimation. They compared each interpolation technique in terms of bias, mean absolute error (MAE), and mean squared error (MSE). For kriging and cokriging, cross validation techniques were used to choose the best **semivariogram** model from among candidate models (spherical, exponential, or Gaussian) and to select a search radius which minimizes the kriging variance. Collins and Bolstad (1996) found evidence that the choice of spatial interpolation technique was influenced by certain a priori data characteristics. The greatest influence on interpolation accuracy comes mainly from data measurement accuracy, data density, data distribution, and spatial variability.

Jimenez (1995) also mentioned various ways to account for spatial variability, each having its own advantages and disadvantages. Jimenez's (1995) focus was how to design an efficient sampling scheme to accurately determine land characteristics over a site. These land characteristics can be attributes such as soil, topography, and rainfall, which can be used when characterizing a hazardous waste site.

Another comparison was accomplished by Journel and Huijbregts (1978). They compared kriging with other weighted methods used in mining applications. They also identified two common criticisms used against kriging and argued why these criticisms are insignificant.

2.2.1 *Knowledge of the Distribution of Natural Resources*

Jimenez (1995) stated that the advantage of knowing something about how the natural resource was caused is that it uses our knowledge about the causes of spatial variability in the natural resource. However, no information is available on the distribution of values within each map unit.

2.2.2 *Using a Single Nearby Value*

The advantage of using a single nearby value is that it uses actual data without making any assumptions about spatial structure (Jimenez, 1995). On the other hand, sudden changes in values between polygon boundaries can affect it, polygons depend on the sample points, polygons can have unusual shapes, and no errors are assumed within polygons.

2.2.3 *Interpolating from Nearby Values*

An advantage of interpolating from nearby values is that it uses more observational data (Jimenez, 1995). Some disadvantages are that this method assumes continuous behavior of the variable, does not account for redundant observations, and has no way to determine the error of the estimate at an interpolated point.

2.2.4 *Inverse Distance Squared (IDA)*

Collins and Bolstad (1996) found that Inverse Distance Squared (IDA) worked well with a priori data, correlation, and variance. They also determined that IDA was

rather robust to the effects of a parameter's range, variance, and correlation with elevation. The elevation correlations they are referring to are those present because temperatures vary with elevation. This could apply to a hazardous waste site when properties are present, such as contaminant concentrations, which vary with elevation. When data were sparse, IDA gave unlikely results. IDA also suffered from discontinuities that resulted in peaks in parameter values. These parameter peaks resulted in unlikely patterns in the contour maps. IDA's one advantage was that it consistently adhered to the original range of data. This is favorable when the data are representative of the surface being interpolated. Where the data are not representative of the surface being modeled, interpolation biases may occur.

2.2.5 *Optimal Inverse Distance (ODA)*

Optimal Inverse Distance (ODA) worked well with a priori data, correlation, and variance. Collins and Bolstad (1996) found that it also appeared to be the most favored method when the data were **isotropic**. Additionally, ODA performed better than kriging based on MAE when the data were isotropic. ODA was rather robust to the effects of a parameter's range, variance, and correlation with elevation. ODA had consistently better results than IDA or kriging. However, ODA's visual representation, when compared to kriging, tended to be less probable. Since the power parameter is chosen based on minimum MAE, ODA is recommended over IDA because ODA's MAE results will always be equal to or better than IDA's MAE results. Given ODA's advantage over kriging for isotropic data, Collins and Bolstad (1996) recommended ODA in cases where **ancillary information** correlated with parameters is not available.

2.2.6 *Splining*

Splining appeared to be most sensitive to a priori data characteristics. Splining is mathematically equivalent to kriging; nevertheless, it gave poor visual and cross validation results. Where data variances were high, splining often interpolated values well outside the observed data range. For lower correlations, splining had very high MAE values. Collins and Bolstad (1996) did not generally recommend splining when the data are irregularly spaced. On the other hand, splines are useful for quickly showing the main feature of the variable, but are not an accurate interpolator. Splining generally has more outliers than kriging and higher MAE values.

2.2.7 *Polynomial Regression*

Polynomial regression had the lowest MAE value and was best at representing the original data; therefore, Collins and Bolstad (1996) concluded that this method was favored over other methods when data correlation was high between parameters and elevation. The only situations where they found MAE values lower for polynomial regression, were when correlations between elevation and parameters were lower than 0.72. When correlations between parameters and elevation were very low (under 0.20) ODA and kriging had lower MAE values. Polynomial regression performance was seemingly unaffected by data range and it produced the most probable visual representation.

2.2.8 *Trend Surface Analysis (TSA)*

TSA's results did not represent the original data well partly because its interpolated parameter range was usually more narrow than the original data. However, TSA was rather robust to the effects of a parameter's range, variance, and correlation with

elevation. TSA tended to capture broad regional trends, but due to bias introduced by multicollinearity, these trends are questionable. TSA occasionally interpolated parameters well beyond the original data range.

2.2.9 *Kriging*

Kriging worked well with a priori data, correlation, and variance. Collins and Bolstad (1996) favored kriging over ODA when data were **anisotropic**. Kriging typically had lower MAE values than IDA did. In addition, kriging had lower MAE values than all other methods when the data were anisotropic. Kriging was rather robust to the effects of a parameter's range, variance, and correlation with elevation. Of the methods compared in this study that did not use elevation as ancillary information, kriging had the most probable visual representation. In some cases, kriging appeared similar to splining, but had lower MAE values than splining. Collins and Bolstad (1996) also found that kriging estimates were better than IDA in some instances. Using available semivariogram modeling tools, data anisotropy can be determined, the best fitting semivariogram model can be selected, and kriged results can easily be obtained. Collins and Bolstad (1996) found that kriging was more complex and only gave marginal benefits over ODA; therefore, justifying the use of kriging over ODA was questionable. However, the greatest advantage of kriging is that the geostatistical process provides the users with greater information about the spatial variability of the regionalized variable of interest via semivariograms and **variogram** surfaces.

Jimenez (1995) pointed out that kriging is mathematically optimal and provides an error estimate. However, some disadvantages Jimenez (1995) stated are that it is computationally intensive, has a problem with the choice of the best spatial area to use

for an estimate, assumes continuous variables, and makes strong assumptions about the spatial structure.

Journel and Huijbregts (1978) stated that kriging provides the best unbiased estimator among linear interpolation methods when a structural analysis is sufficient to correctly determine the basic structural characteristics of the data. They also established that this property is obvious and not shared by other common linear estimation methods used in mining applications such as polygons of influence, inverse distances, inverse-square distances, and least-square polynomials.

One criticism used against kriging is that there are insignificant advantages in estimation variance provided by kriging when compared to the other methods (Journel and Huijbregts, 1978). However, kriging is optimal; therefore, small changes in weights around optimal values will only cause a second-order increase in the estimation variance. In addition, most of the other common methods are only sufficient for particular data configurations. For instance, if the data are anisotropic, a modified version of inverse-square distances may be highly effective. If the data are isotropic, least-square polynomials may be preferred. However, since the data configurations may vary greatly throughout a site, kriging is the surest method to use (Journel and Huijbregts, 1978).

Another criticism often used against kriging, which was also pointed out by Jimenez (1995), is that it is costly to implement in terms of sampling and computer time (Journel and Huijbregts, 1978). However, the computational cost of kriging is irrelevant when compared to the cost of taking a few samples at the site. Kriging can be computationally expensive if a theoretically optimum solution is required. However, the optimum solution is not practical because the computational costs would not be

competitive with the other methods. Consequently, Journel and Huijbregts (1978) proposed that establishing a "kriging plan" could reduce the cost and dimension of kriging. Details on how to elaborate a kriging plan are beyond the scope of this thesis. Discussion on how to elaborate a kriging plan can be found in Journel and Huijbregts (1978).

2.2.10 Cokriging

Cokriging produced unlikely visual representations. That is, unlikely patterns were produced in contour maps when data variance was high or correlation between parameters and elevation was low. On the other hand, Collins and Bolstad (1996) encountered more consistent patterns in contour maps when data variance was low and correlation between parameters and elevation was high. Since cokriging's elevation component is not significant, it strongly resembles kriging when elevation and parameters are not correlated. However, they found cokriging more accurate than kriging in predicting parameters when elevation correlations were significant. Collins and Bolstad (1996) did not recommended cokriging when correlations between parameters and elevation are below 0.72.

3. Design Decisions

3.1 Spatial Interpolation Technique Selection

Based on the literature review, kriging was chosen as the spatial interpolation technique for Sim-Site. One feature of kriging, pointed out in Section 2.2.9, is that it works well with clustered data. However, this feature could not be used to justify kriging because the data used to obtain a sample result in Sim-Site is fairly equally distributed. That is, the data is already on a grid created by GMS.

Another feature of kriging is that it was favored over other techniques when the data were anisotropic. However, the anisotropy of the "real" site can vary from one scenario to the next. Consequently, this feature did not have a significant influence in the selection of a technique. If the data are isotropic, kriging will still provide an optimal estimate although other techniques may be favored that are less complex and computationally expensive.

The greatest advantage of kriging, pointed out in Section 2.2.9, is that it provides the users with greater information about the spatial variability of the regionalized variable of interest via semivariograms and variogram surfaces. However, this information is not currently conveyed to the Sim-Site user. The Sim-Site developer may decide to use this information in subsequent versions of Sim-Site to provide the user with variograms or to use this during **cross validation**. Cross validation is similar to a dress rehearsal: it is intended to detect what could go wrong, but it does not ensure that the show will be successful (Deutsch and Journel, 1992).

The most influential feature used to choose kriging over the other methods, which was pointed out in section 2.2.9, was that kriging is optimal. Most of the other common

methods are only sufficient for particular data configurations. Consequently, kriging is the surest method to use because the data configurations may vary greatly throughout a site.

3.1.1 *Kriging Technique Selection*

Now that kriging was selected as the linear interpolation technique for Sim-Site, the type of kriging must be selected. Numerous kriging techniques with varying degrees of sophistication have been developed, but the two most commonly used techniques are ordinary kriging and universal kriging (Brigham Young University, Feb. 17, 1998). Of these, ordinary kriging was chosen because it is considered by Deutsch and Journel (1992) as the anchor algorithm of geostatistics. Additionally, kriging routines were published by Deutsch and Journel (1992), which includes FORTRAN code for ordinary kriging.

All variations of kriging are elaborations on the basic linear regression algorithm and corresponding estimator:

$$[Z_{SK}^*(u) - m(u)] = \sum_{\alpha=1}^n \lambda_{\alpha}(u) [Z(u_{\alpha}) - m(u_{\alpha})] \quad \text{Equation 1}$$

where $Z(u)$ is the random variable model at location u , the u_{α} 's are the n data locations, $m(u) = E\{Z(u)\}$ is the location-dependent expected value of random variable $Z(u)$, $\lambda_{\alpha}(u)$ are the weights determined to minimize error variance, and $Z_{SK}^*(u)$ is the linear regression estimator, also called the "simple kriging" estimator (Deutsch and Journel, 1992).

3.1.2 Ordinary Kriging

Ordinary kriging filters the mean from the "simple kriging" estimator (Equation 1) by requiring that the kriging weights sum to one. This results in the following ordinary kriging estimator:

$$Z_{OK}^*(u) = \sum_{\alpha=1}^n w_{\alpha}(u) Z(u_{\alpha}) \quad \text{Equation 2}$$

and the stationary ordinary kriging system:

$$\begin{cases} \sum_{\beta=1}^n w_{\beta}(u) C(u_{\beta} - u_{\alpha}) + \mu(u) = C(u - u_{\alpha}), \alpha = 1, \dots, n \\ \sum_{\beta=1}^n w_{\beta}(u) = 1 \end{cases} \quad \text{Equation 3}$$

where $C(h)$ is the stationary covariance between any two random variables $Z(u)$, the $w_{\beta}(u)$'s are the ordinary kriging weights, and $\mu(u)$ is the Lagrange parameter associated with the constraint $\sum_{\beta=1}^n w_{\beta}(u) = 1$. Ordinary kriging is the same as simple kriging except that simple kriging uses a constant mean while ordinary kriging uses a location-dependant mean. Since kriging is a rather complex interpolation technique and includes numerous options, a complete description of kriging is beyond the scope of this thesis. More in-depth information on kriging can be found in Deutsch and Journel (1992) and Journel and Huijbregts (1978).

The first step in ordinary kriging is to construct a variogram from the scatter point set to be interpolated (Brigham Young University, Feb. 17, 1998). A variogram consists of two parts: an experimental variogram and a model variogram. The experimental variogram is found by calculating the variance (γ) of each point in the set with respect to each of the other points and plotting the variances versus distance (h) between the points (Figure 1).

Once the experimental variogram is computed, the next step is to define a model variogram. A model variogram is a simple mathematical function that models the trend in the experimental variogram (Figure 1).

As can be seen in Figure 1, the shape of the variogram indicates that at small separation distances, the variance is small. In other words, points that are close together have similar values. After a certain level of separation, the variance becomes somewhat random and the model variogram flattens out to a value corresponding to the average variance. Once the model variogram is constructed, it is used to compute the weights used in kriging.

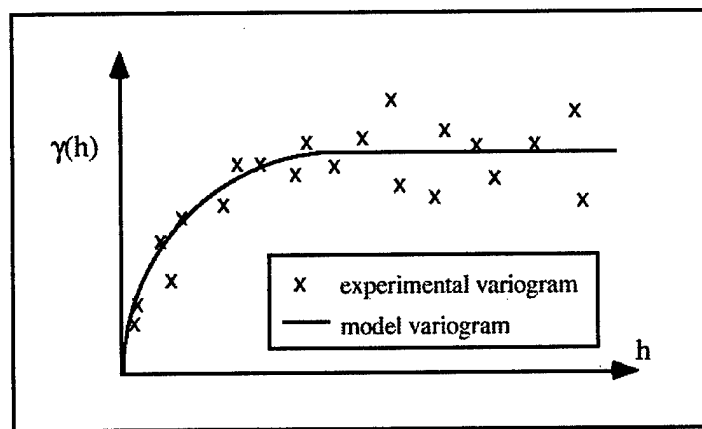


Figure 1 Variogram Used in Kriging

3.2 Programming Language Selection

Object oriented programming uses three models to describe a system: the **object model**, **dynamic model**, and **functional model**. The object model describes the objects and their relationships, the dynamic model describes the interactions among the objects, and the functional model described the data transformations. Each model describes one aspect of the system and all three are necessary for any large system (Rumbaugh et al., 1991). However, only the object model was used to develop Sim-Site because Sim-Site

is not a large system and the object model is considered the most significant of the three models (Rumbaugh et al., 1991). Therefore, only the object model is included in the software specification in Appendix A.

In order to implement the design, a programming language had to be selected that can map to an object-oriented design. According to Rumbaugh et al. (1991), an object-oriented language supports **objects** (combining data and operations), **polymorphism** at run-time, and **inheritance**.

Microsoft Visual Basic 6.0 (VB6) was the language of choice to develop the simulator software. VB6 is a fast and easy way to create applications for Microsoft Windows (Microsoft Corporation, 1998). It provides a complete set of tools to simplify rapid application development. Rather than writing numerous lines of code to describe the appearance and location of interface elements, prebuilt objects can be added into place on the screen to create an effective user interface.

Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows Graphical User Interface (GUI). The power of the language allows almost anything to be accomplished that can be accomplished using any other Windows programming language (Microsoft Corporation, 1998).

The Visual Basic programming language is not unique to Visual Basic. The Visual Basic programming system, Applications Edition included in Microsoft Excel, Microsoft Access, and many other Windows applications uses the same language (Microsoft Corporation, 1998). The Visual Basic Scripting Edition (VBScript) is a widely used scripting language and a subset of the Visual Basic language.

Visual Basic also has some powerful tools. Data access features allow the creation of databases, front-end applications, and scalable server-side components for most popular database formats. **ActiveX** technologies allow the use of functionality provided by other applications, such as Microsoft Word word processor, Microsoft Excel spreadsheet, and other Windows applications. Internet capabilities make it easy to provide access to documents and applications across the Internet or Intranet from within an application, or to create Internet server applications. The finished application is a true .exe file that uses a Visual Basic Virtual Machine that can be freely distributed.

Visual Basic was also chosen because it can be used for object-oriented programming. Objects are central to Visual Basic programming (Microsoft Corporation, 1998). Forms and controls are objects. Databases are objects. The following considerations apply to implementing an object-oriented design in an object-oriented language: **class** definitions, creating objects, calling operations, using inheritance, and implementing associations (Rumbaugh et al., 1991).

3.2.1 *Class Definitions*

The first step in implementing an object-oriented design is to declare object classes. In Visual Basic, a new public class can be defined by choosing *Add Class Module*, *Add User Control*, or *Add User Document* from the *Project* menu. Other choices on the *Project* menu allow the addition of objects that can be used within an application, but only user controls, user documents, and class modules can define public classes.

Each public class added will be the blueprint for one kind of public object in the object model. To determine how objects will be created from the class, a class name can

be provided, interfaces for the class can be defined, and the Instancing **property** (or the *Public* property, in some cases) can be set.

Sim-Site contains many class definitions. For instance, the *Well* class is a public class that is used to hold an instance of a well. The interfaces for this class are those that either return or assign a property value. That is, another object or program can retrieve or set the well's properties (e.g., the well's location).

3.2.2 *Creating Objects*

Object-oriented languages create new objects in one of two ways. Some languages have classes that are full objects in their own right. Other languages have special operations that create new objects. Objects in Visual Basic are created from classes; thus, an object is said to be an instance of a class. The class defines an object's interfaces, whether the object is public, and under what circumstances it can be created. Descriptions of classes are stored in type libraries, and can be viewed with object browsers.

An object is created in Sim-Site by creating an instance of a class using the VB6 *New* keyword. For instance, the *GMSMap* class creates a *Grid* object from the *Grid* class by declaring the following:

```
Private mobjGrid As New Grid
```

The variable *mobjGrid* holds the instance of a *Grid*.

3.2.3 *Calling Operations*

In most object-oriented languages, each operation has at least one implicit argument, the target object, indicated with a special syntax. Operations may or may not have additional arguments. Some languages permit a choice between passing arguments as read-only values or as references to values that can be updated by a procedure.

Visual Basic's calling operations use the target object followed by **methods** defined by that object. One statement could contain several objects depending upon the object hierarchy. Also, Visual Basic permits a choice between passing values by value or by reference using the keywords **ByVal** (read-only) and **ByRef** (can be updated by the procedure) in the procedure's argument list.

An example of a calling operation in Sim-Site is:

```
mobjGrid.GridRowSpace(Row) = Val(GMSFilebcf.GetNextField)
```

The target object is the grid and the method is the row spacing of the grid. An example of a procedure declaration that shows passing parameters by value or by reference is:

```
Public Sub ksol(ByVal nright As Integer, _  
                ByVal neq As Integer, _  
                ByVal nsb As Integer, _  
                a() As Double, _  
                r() As Double, _  
                s() As Double, _  
                ByRef ising As Integer)
```

The calling operation for the above procedure declaration is:

```
ksol nright:=1, _  
    neq:=neq, _  
    nsb:=1, _  
    a:=a, _  
    r:=r, _  
    s:=s, _  
    ising:=ising
```

The variables that can be changed by the procedure *ksol* are *a*, *r*, *s*, and *ising*. The first three variables mentioned are arrays that can only be passed by reference although the called procedure might not change their values. The *ByRef* keyword is the default when neither *ByRef* nor *ByVal* are used.

3.2.4 Using Inheritance

Most object-oriented programming systems provide polymorphism through inheritance. Visual Basic does not use inheritance to provide polymorphism. In fact,

VB6 does not support single and multiple inheritance in the classic sense of the term, but you can subclass existing objects with it (Swartzfager et al., 1999). Visual Basic provides polymorphism through multiple ActiveX interfaces. In the Component Object Model (COM) that forms the infrastructure of the ActiveX specification, multiple interfaces allow systems of software components to evolve without breaking existing code. An interface is a set of related properties and methods. Much of the ActiveX specification is concerned with implementing standard interfaces to obtain system services or to provide functionality to other programs.

Since VB6 does not support inheritance, neither does Sim-Site. However, Sim-Site does use polymorphism. An example is the *Draw* method contained in the *GMSTMap* class. This polymorphic method draws a drawing object (text, rectangle, oval, or line), well, or grid frame on the site image. The *Draw* method works by first recognizing the object to draw, and then drawing the object on the site image.

3.2.5 *Implementing Associations*

The only association used in Sim-Site involves the sample methods and the parameters. A parameter is determined by one or more sampling methods and a sampling method determines one or more parameters. The association was not modeled as a class, but rather as two distinct arrays, one in each class. For instance, a sample method has an array containing keys (similar to pointers) to the parameters it can determine. It was modeled like this because some sample methods can determine the same parameters and some parameters can be determined by the same sampling methods.

3.3 *GMS Image Files*

Image files are used in conjunction with Tag(ged) Image File Format (TIFF) files, which have been previously imported to GMS and registered (Brigham Young University, Feb. 14, 1998). They include the name of the TIFF file, the registration points, and the bounds of the clipping window. The TIFF files are required to be in a format called the Apple Macintosh PackBits scheme, a simple byte-oriented run-length scheme (Birse and Marinkovich, 1996). GMS image files are saved in ASCII format; therefore, the text from the file can be read directly by Sim-Site. On the other hand, the TIFF image is stored in a binary format, requiring the data to be read in one byte at a time.

Sim-Site is required to display the TIFF image originally used by GMS; however, TIFF images cannot be directly imported into a Visual Basic *PictureBox* control. A *PictureBox* control can only display a graphic from a bitmap, **icon**, or **metafile**, as well as enhanced metafile, JPEG, or GIF files. Therefore, another method was needed to load the image into the *PictureBox* control.

The first method used to load the image involved the creation of a procedure to read the TIFF file in directly and paint the *PictureBox* one point at a time. In order to develop the procedure, the TIFF file format was obtained (Adobe, 1992). An alternative to this would be to have the user convert the image to a bitmap using an image editor. However, GMS uses image files in conjunction with TIFF files, which have been previously imported to GMS and registered. The image files include the name of the TIFF file, the registration points, and the bounds of the clipping window. Thus, having the user convert the image would be impractical.

Writing the image directly to the *PictureBox* control using the new routine took too long (a few minutes); therefore, an alternative method had to be devised. After creating the routine to read in a TIFF file, an alternative method was discovered using an existing control called the Wang *Image Edit* control. However, in order for the image to be loaded properly, a program called Imaging for Windows needs to be installed on the user's computer. Imaging for Windows was developed by Eastman Software for Microsoft and is currently included as an accessory in every copy of Windows 95, Windows 98, and Windows NT. Imaging for Windows is also provided free of charge and can be downloaded from Eastman Software's web site (http://www.eastmansoftware.com/products/ImagingPro/pr_pro_i4w.htm).

The Wang *Image Edit* control adds imaging and image annotation functions to applications that support 32-bit OLE controls. It enables end users to display, annotate, manipulate, and manage image files. The *Image Edit* control allows end users to display image files of various types. Specifically, the control supports the following file types: AWD, BMP, DCX, JPG, PCX, TIFF, and XIF. Once the TIFF image is loaded into the *Image Edit* control, it can be saved as a bitmap using the control's *SaveAs* method. Currently, Sim-Site saves the file as "temp.bmp". This file can then be loaded into a control that can read bitmap files. Sim-Site reads the newly created bitmap file into a *PictureClip* control so that it can be clipped according to the clip region set in the GMS image file when registering the image. Then this clipped picture has to be loaded into the *PictureBox* control to complete the loading. A *PictureBox* control has to be used to display the image because it is one of only three controls that can be used as a container for other objects. The other two controls are the *Frame* and *SSTab* controls. In Sim-Site,

for example, objects such as wells and sampling points are contained in the *PictureBox* control. This allows for easier manipulation and placement of the objects within the image of the site.

4. Implementation

4.1 GMS Files

Most of the files used by GMS use a "card" type format. With this format, the components of the file are grouped into logical groups called "cards." The first component of each card is a short name that serves as the card identifier. The remaining fields on the line contain the information associated with the card. In some cases, such as lists, a card can use multiple lines. Additional information on GMS file formats can be found in *GMS 2.1 File Formats* (Brigham Young University, Feb. 14, 1998).

Most of the GMS files are stored in ASCII format, making them very easy to read by Sim-Site. Additionally, the card type format allows the information from the files to be easily extracted. The following sections list the GMS files used by Sim-Site and how they are applied. Many other GMS files are available that are not used in the current version of Sim-Site and are most likely not needed for future versions. Section 4.1.4 lists a few GMS files that may be useful in future versions of Sim-Site.

4.1.1 Tabular Scatter Point Files (*.mds)

When scatter point sets are saved by GMS, the scatter point coordinates are saved in one file and the values at the scatter points are saved in a data set file. When importing scatter point data into GMS for the first time, this format may be used if desired. However, scatter point sets are typically created by importing from a text file using the *Import* command in the *File* menu in GMS.

Sim-Site does not read GMS scatter point files. Instead, Sim-Site has an option to create a scatter point file from the samples taken by the user. When the Sim-Site user chooses *Display | Sample Results* from the menu, the sample results are displayed. The

user can then click on the *Save Scatter Point* button to save the values in a scatter point text file. This allows the user to easily add these scatter points to the conceptual model built from the samples. The user then creates a mathematical model from this conceptual model, which is compared to the "real" site at the end of a Sim-Site characterization exercise (i.e., all seasons have been completed).

4.1.2 Map Files (*.map)

Map files are used to store feature object and drawing object data (Brigham Young University, Feb. 14, 1998). The map file includes not only the geometric description of such objects, but also the attributes associated with each object. GMS map files are saved in ASCII format; therefore, the text from the file can be read directly by Sim-Site.

Feature objects include points, nodes, vertices, arcs, and polygons. Points in a local source/sink coverage normally define wells, which are used by Sim-Site. Nodes, vertices, arcs, and polygons normally specify items such as head boundaries and zones of hydraulic conductivity, which are not used by Sim-Site because the user determines this information. The user only needs to know where the existing wells are for sampling purposes.

Drawing objects include rectangles, ovals, lines, and text. These are all imported and used by Sim-Site because they are considered aids to understanding the layout of the site being characterized. These objects can point to such features as rivers, possible boundaries, and suspected contaminant location.

The map file also includes the grid frame. Results obtained through running the **Modular Finite-Difference Ground-Water Flow Model (MODFLOW)** are confined to

this grid frame. Therefore, the grid frame is used by Sim-Site to let the user know the sampling boundaries. That is, the user will not be allowed to obtain a sample outside the grid boundaries because no useful values are present.

4.1.3 Image Files (*.img)

Image files are used in conjunction with TIFF files, which have been previously imported to GMS and registered (Brigham Young University, Feb. 14, 1998). They include the name of the TIFF file, the registration points, and the bounds of the clipping window. Sim-Site currently uses the image files to import the site picture for each scenario.

4.1.4 Other Files

Table 1 lists the files available from GMS, not currently used by Sim-Site, which could be useful in future versions of Sim-Site.

Table 1 Standard GMS File Types Not Currently Used by Sim-Site

File Type	File Extension	Contents
2D Scatter Point File	.xy	One or more sets of 2D scatter points. Only the point locations are saved in this file. The values associated with the points are stored in data set files.
3D Scatter Point File	.xyz	One or more sets of 3D scatter points. Only the point locations are saved in this file. The values associated with the points are stored in data set files.
Data Set File	.dat	Scalar or vector data associated with a particular object (mesh, grid, scatter point set, etc.). The file can contain one or multiple data sets. The data may be steady state or transient. For each data set or time step of a data set, there is one value for each node, cell, or vertex of the object the data set is associated with.

The data set file (*.dat) contains data sets or time steps of data sets, associated with 2D or 3D scatter point files, which could be used as possible input to Sim-Site. The

data could be anything from a particular contaminant to a velocity. The most likely use of the data set file is to import kriging variogram parameters into Sim-Site. If variograms have been defined for a data set or time step of a data set within GMS, the variograms are saved in the data set file.

4.2 *MODFLOW Files*

GMS includes a comprehensive graphical interface to the groundwater model called MODFLOW. MODFLOW is a 3D, cell-centered, finite difference, saturated flow model developed by the United States Geological Survey (Brigham Young University, Feb. 17, 1998). MODFLOW can perform both steady state and transient analyses and has a wide variety of boundary conditions and input options.

GMS supports MODFLOW as a pre- and post-processor. The input data for MODFLOW are generated by GMS and saved to a set of files. These files are then read by MODFLOW when MODFLOW is executed. The output from MODFLOW is also saved to a set of files, which can then be imported to GMS for post-processing.

4.2.1 *Super Files (*.sup)*

A MODFLOW super file allows all of the packages involved in a MODFLOW simulation to be input at once. It is used by GMS and by the version of MODFLOW distributed with GMS. It is not an original MODFLOW file. The file contains a list of the package files used in the simulation. It also contains data needed by GMS that are not contained in any of the package files such as the layering and orientation of the 3D grid. The MODFLOW super files are not currently used by Sim-Site; however, these files can be used as an alternate to the Output Listing file discussed in the next section.

4.2.2 *Output Listing (*.out)*

The Output Listing is a text file containing a summary of the output data. These files are generated by MODFLOW as the solution is computed. The file names for each MODFLOW package are included in the Output Listing file. The full path of the file is included in the output, not just the file name; therefore, it was favored over the MODFLOW Super File to retrieve file names.

Sim-Site currently uses the Output Listing files to obtain the Block-Centered Flow Package file name, the grid cells that define the active coverage, and the number of rows and columns in the output grid. The active coverage cells are those cells that are used in the MODFLOW simulation to define the no-flow boundaries for the site. In other words, MODFLOW will not generate values for the cells outside this area. The person who creates the scenario for Sim-Site will identify the output file in the scenario file (*.snr) used by Sim-Site.

4.2.3 *Basic package (*.bas)*

The Basic package includes data defining fundamental program options such as the computational time intervals (stress periods), an array defining which cells are inactive and which cells have constant heads, an array of starting head values for a transient simulation, and an array defining which of the other packages are to be used.

The Basic Package is not currently used by Sim-Site; however, it can be used as an alternative to obtain the active coverage and the number of rows and columns in the output grid.

4.2.4 Block-Centered Flow Package (*.bcf)

The Block-Centered Flow package computes the conductance between each of the grid cells and sets up the finite difference equations for the cell to cell flow. It also computes the terms that determine the rate of movement of water to and from storage.

Sim-Site currently uses this file to obtain the spacing between columns and rows and the hydraulic conductivities for each grid cell.

4.2.5 Other Files

Table 2 lists other files available from MODFLOW, not currently used by Sim-Site, which could be useful in future versions of Sim-Site. These may be incorporated as more sampling parameters are added.

Table 2 MODFLOW Packages Not Currently Used by Sim-Site.

Package Name	File Extension	Description
River Package	.riv	Simulates river type boundary conditions.
Recharge Package	.rch	Simulates recharge to the groundwater from precipitation.
Well Package	.wel	Simulates injection/extraction wells.
Drain Package	.drn	Simulates drain type boundary conditions.
Evapotranspiration Package	.et	Simulates the effect of evapotranspiration in the vadose zone.
General Head Boundary Package	.ghb	Simulates a general purpose head-dependent source/sink. Commonly used to simulate lakes.
Stream/Aquifer Interaction Package	.str	Simulates the exchange of water between the aquifer and surficial streams. Includes routing and automatic computation of stage.
Horizontal Flow Barrier Package	.hfb	Simulates the effect of horizontal flow barriers such as sheet piles and slurry trenches.
Time Variant Specified Head Package	.chd	Simulates specified head boundary conditions where the head is allowed to vary with time.

4.3 *Kriging Program Translation*

The ordinary kriging procedures used in Sim-Site were translated almost directly from existing FORTRAN code to VB6 with a few minor variations. The FORTRAN code was obtained from the **Geostatistical Software Library** (GSLIB) and User's Guide (Deutsch and Journel, 1992).

The FORTRAN kriging procedure uses a data file to input initial parameters. This data file includes file names for the sample data, debugging, and output files. Sim-Site does not directly use a data file to retrieve sample data to be kriged because the data was already loaded when the scenario was opened. Therefore, the VB6 procedure was modified to accept the initial data as input, which includes the sample grid, estimation block parameters, and various kriging parameters. The output and debugging files are still included in the VB6 procedure, which are passed as parameters by the calling procedure. Additionally, the FORTRAN procedure writes output to the screen, whereas the VB6 procedure writes the same output to the immediate window. The difference here is that the user will not see this output but the programmer will. This was left in the Sim-Site as a potentially useful debugging tool.

The FORTRAN procedure contained a sort routine that used several temporary variables. This routine sorted the samples found at that point in the code by increasing order of distance. This routine was somewhat confusing and ended up being the main cause of an error after the translation to VB6 was done. Therefore, this routine was replaced with another sort routine. The VB6 procedure worked correctly immediately after the change was made.

During the translation, most of the same variable names that were used in the FORTRAN code were used in the VB6 code. Three exceptions were *is*, *in*, and *id*. These are reserve words in VB6; therefore, they were modified as *iss*, *inn*, and *idd* respectively. The use of the same variable names from the FORTRAN code made the readability of the code rather poor compared with the rest of the code in Sim-Site. However, this was done to aid the programmer in debugging or when code modifications need to be done. Additionally, many of the original comments were used, which allows the programmer to easily refer back to the correct location in the FORTRAN code.

The FORTRAN code contained a few undesirable structures. For instance, *go to* was used in several locations to exit out of *do* loops. All of these *go to* statements were eliminated from the VB6 code and replaced with *For...Next* loops and *If ... Then ... Else ... End If* structures. Consequently, one of the *go tos* from the sorting routine mentioned earlier, jumped out of two *do* loops. The intention of this *go to* was misinterpreted and was partially responsible for the error with the sort routine. In addition, one major undesired item was the use of static array dimensions. The FORTRAN code first defined the maximum values for certain data and then dimensioned the arrays to their maximum value. This was an unwanted feature for Site-Site because Sim-Site already uses a fair amount of memory to load the site image and objects on that image. Therefore, all of the arrays in the VB6 version are now dynamic based on the parameters passed to the kriging procedure. However, this places no bounds on the dimensions of the data being sent and could cause an overflow in extreme circumstances. For instance, the variable *nn* is defined as an *Integer* type. If a value greater than 179 is sent for the maximum number of data points used in one kriging system, an overflow would occur if a check was not

done first. The overflow is caused by multiplying two *Integer* types together that result in a value greater than 32,767. Consequently, a conditional statement was added to the kriging procedure to prevent an overflow under these circumstances.

4.4 *Comparison of Visual Basic and FORTRAN Kriging Program*

To demonstrate that the converted kriging routines were correctly translated, a couple of tests were conducted to compare the output of both programs. Both of the tests consisted of a 6 x 6 grid of data where the data were evenly spaced ten units apart on the grid. The output grid in all cases was a 5 x 5 grid constructed so that it was small enough not to cover any existing data points. This would be indicative of a normal situation in Sim-Site, where the estimation block is smaller than the node spacing of the real site. This smaller grid was placed in the center of the gridded data with a kriging search radius of 20. In addition, the minimum and maximum data values needed for kriging were set at four and eight respectively. Figure 2 represents the layout of the testing scenarios. The 36 small circles are the data points, the small square in the middle is the solution grid (estimation block), and the circle is the search radius.

In the first test, all of the 36 data values were set to 1.0. The expected value in this case was 1.0. Both programs were run on the same data and both calculated 1.000 as expected.

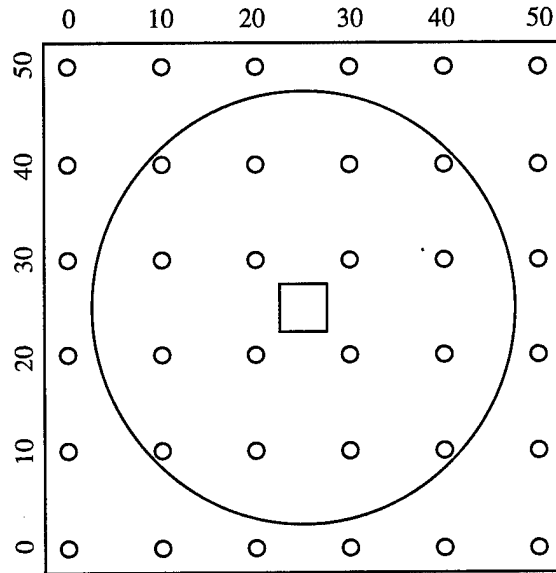


Figure 2 Kriging Comparison Testing Layout

In the second test, the values were set to decrease from left to right in a linear fashion starting from 6.0 to 1.0 as shown in Figure 3. The value of this test was expected to be close to 3.5. Both programs were run and produced slightly different results. The translated VB6 program calculated 3.5000 while the original FORTRAN program calculated 3.5063. This is only a 0.18 percent difference in values, which is insignificant considering the uncertainties that are intrinsic to sampling methods.

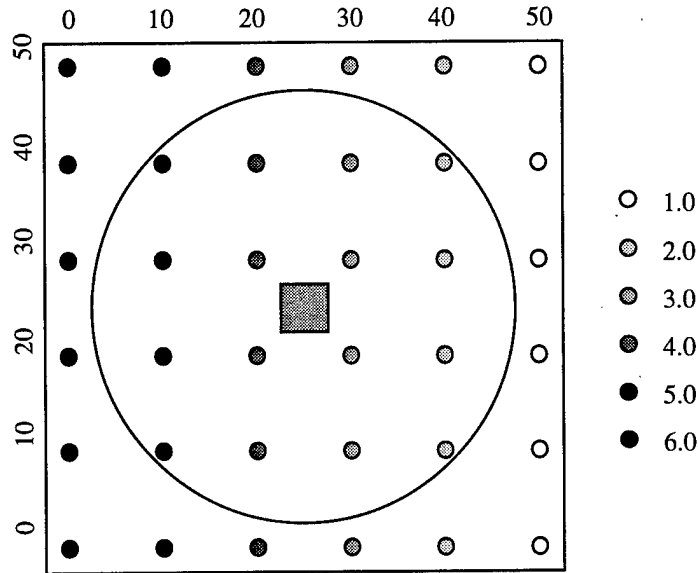


Figure 3 Kriging Comparison Test 2

After comparing the output from both programs, the only significant difference in estimation found was in block (3,3) of the small grid. Note that block (3,3) is in the direct center of the data; therefore, it is suspected that the difference was caused by which particular data points were selected when determining the value at (3,3). That is, the program would first pick the four closest data points, all of which are the same distance away from the point being determined. These points are (20,20), (30,20), (20,30), and (30,30), all of which are 7.07 units away from the point being determined. These points are the circled data points in Figure 4.

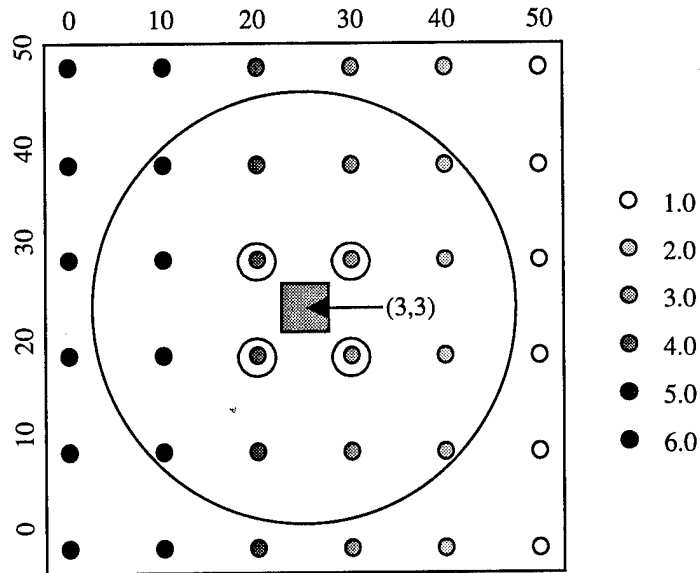


Figure 4 Block (3,3) First Selected Points

Then four more points have to be selected to get to the maximum of eight. However, there are now eight data points exactly 15.81 units from the point being determined. Therefore, the outcome will be affected by which four are picked. If more are picked on the higher value side, the resulting value would most likely be higher. The possible data points that could be picked are circled in Figure 5

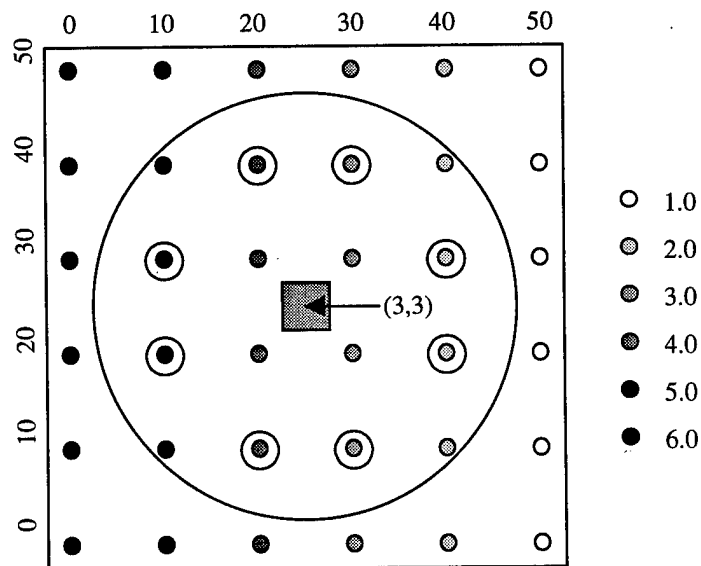


Figure 5 Block (3,3) Possible Choices for Other Four Data Points

The following are segments of the output from both programs:

FORTRAN Program

```

BLOCK:          3          3

Lagrange multiplier: -7.855338752269746E-001
BLOCK EST: x,y,vr,wt
  20.000      20.000      4.000      .171
  20.000      30.000      4.000      .171
  30.000      20.000      3.000      .171
  30.000      30.000      3.000      .171
  10.000      20.000      5.000      .079
  10.000      30.000      5.000      .079
  20.000      10.000      4.000      .079
  40.000      30.000      2.000      .079
est          3.657107
estv         9.441503

```

Visual Basic Program

```

BLOCK:          3          3

Lagrange multiplier: -0.785533905932738
BLOCK EST: x,y,vr,wt
  20.000      20.000      4.000      .171
  20.000      30.000      4.000      .171
  30.000      20.000      3.000      .171
  30.000      30.000      3.000      .171
  10.000      30.000      5.000      .079
  20.000      40.000      4.000      .079
  30.000      10.000      3.000      .079
  40.000      30.000      2.000      .079
est          3.500000
estv         9.441504

```

The FORTRAN program came up with an estimate of 3.6571 and the VB6 program came up with 3.5000 for block (3,3). Since this block was in the middle, the expected value was 3.500.

Figure 6 and Figure 7 show the data points picked by each program. It is apparent that the FORTRAN program picked higher values that caused the higher estimate.

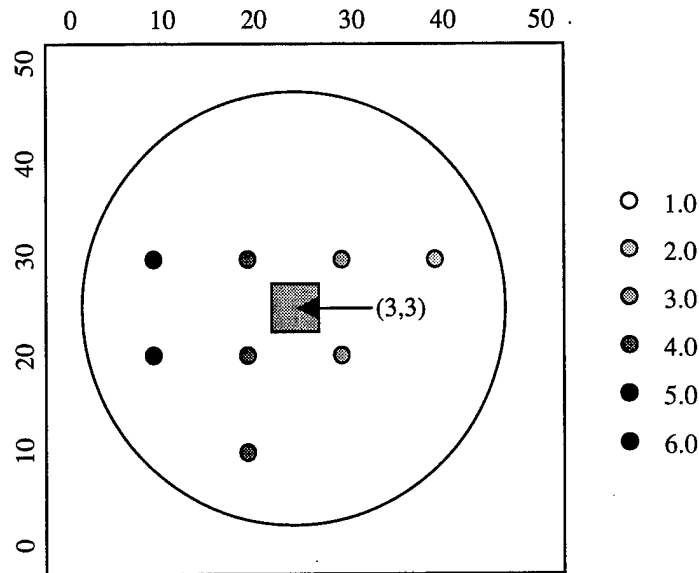


Figure 6 Data Points Picked by FORTRAN Program

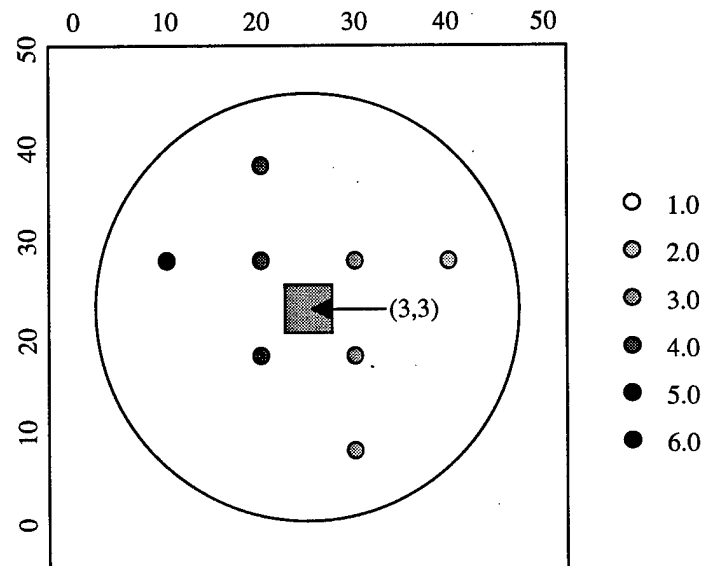


Figure 7 Data Points Picked by Visual Basic Program

4.5 Visual Basic Kriging Program Study

The converted VB6 kriging program now needs to be checked on more heterogeneous data to determine its usefulness to Sim-Site. That is, will it give consistent results under various data configurations? To test this, sample data provided by GSLIB

was used. This sample data contained 2,500 data points on a 50 x 50 uniform grid, which is comparable to the grid used by Sim-Site. Each grid cell is a one unit square. Figure 8 contains a graphical representation of this data. To simulate a sample's estimation block, a sample radius of 0.5 units was used to ensure that no more than two data points are in the estimation block.

To simplify the kriging process, a 23 x 23 portion of the grid in Figure 8 was used for all of the simulations that follow. This grid is shown in Figure 9.

Sample Gridded Data

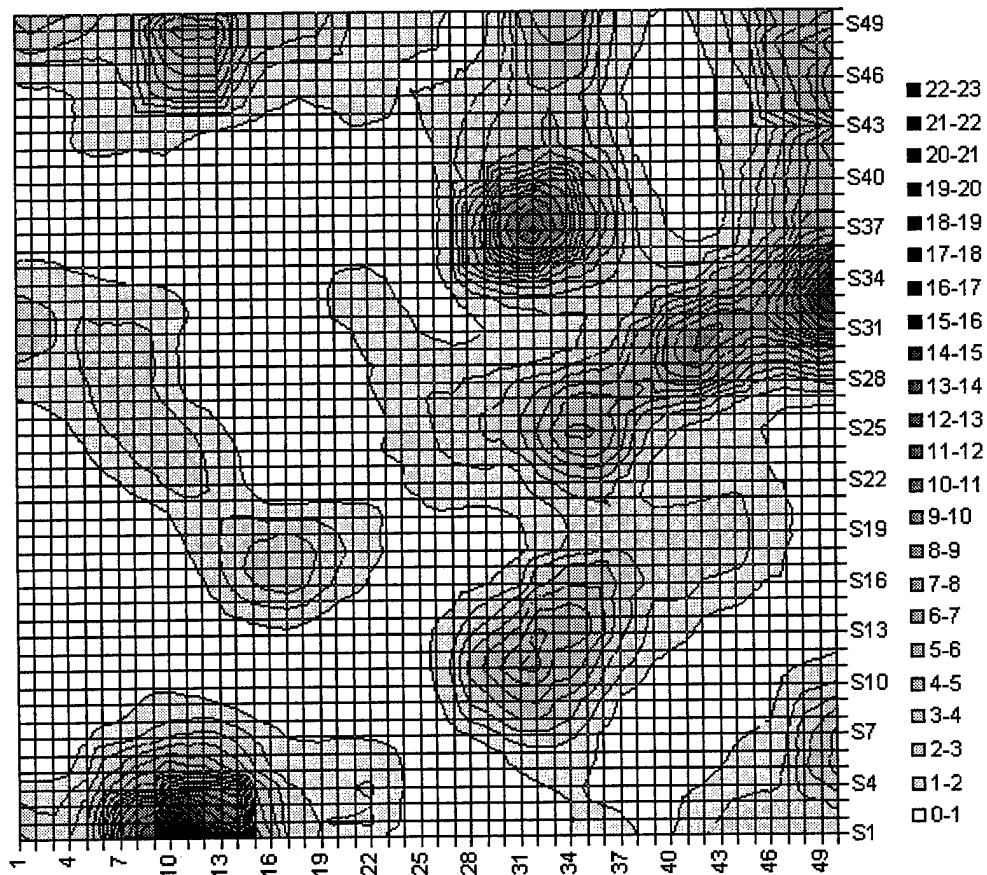


Figure 8 Sample Gridded Data from GSLIB

Three locations were selected for the simulations based on the features present in the data. The first location was picked in a region where the values were rapidly changing. The second location was chosen because it was close to the edge of the data. The third location was chosen because it was in a more or less constant data range. The three locations are represented by circles on Figure 9.

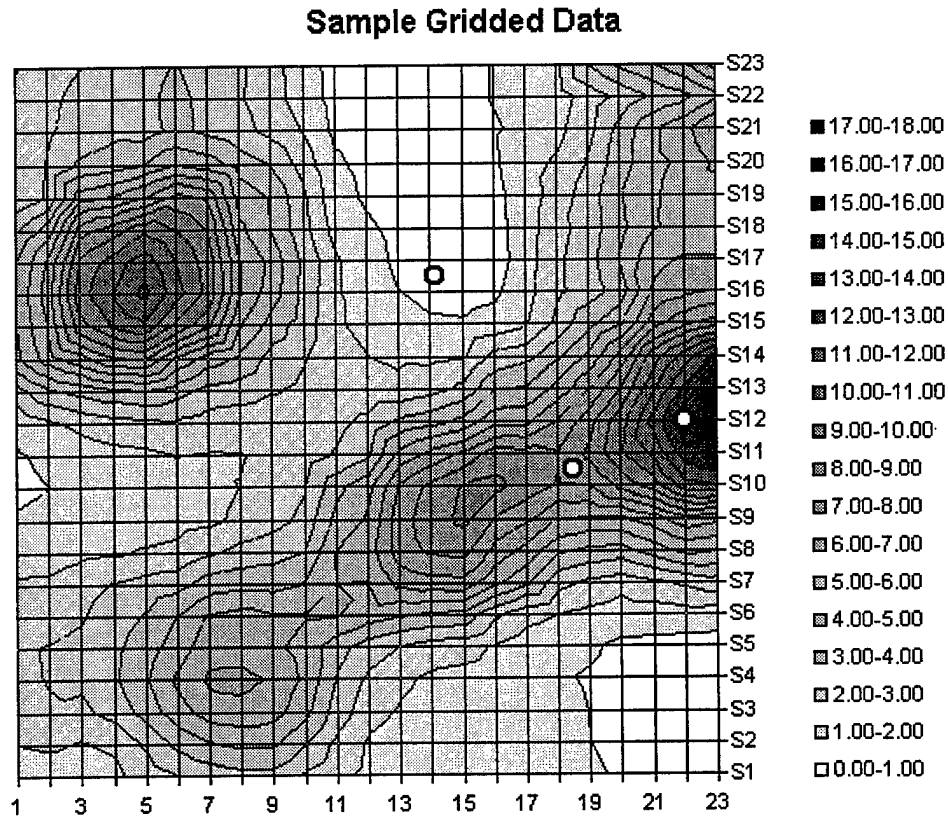


Figure 9 Grid Locations Used for Simulations

In the first simulation, location (18.5, 10.5) was used. This location was where the data values were rapidly changing in almost all directions. The purpose of this simulation was to evaluate the performance of the kriging algorithm with different sized solution grids, which are the grids of the estimation block, and to evaluate what happens when the number of data points varies. Figure 10 contains a graph of this simulation.

It is clear from Figure 10 that the grid size has little effect on the value; however, the number of data points used for kriging does. The average of the four nearest points, indicated by the dark line at value 9.300, was used as a comparison. However, not shown are the minimum of the four nearest values, which was 8.91, and the maximum, which was 10.21. The values obtained by kriging ranged from 9.200 to 9.570 and are well within this range. If the first value is excluded, then the maximum is 9.404 and the entire data range only varies by about 2.2 percent.

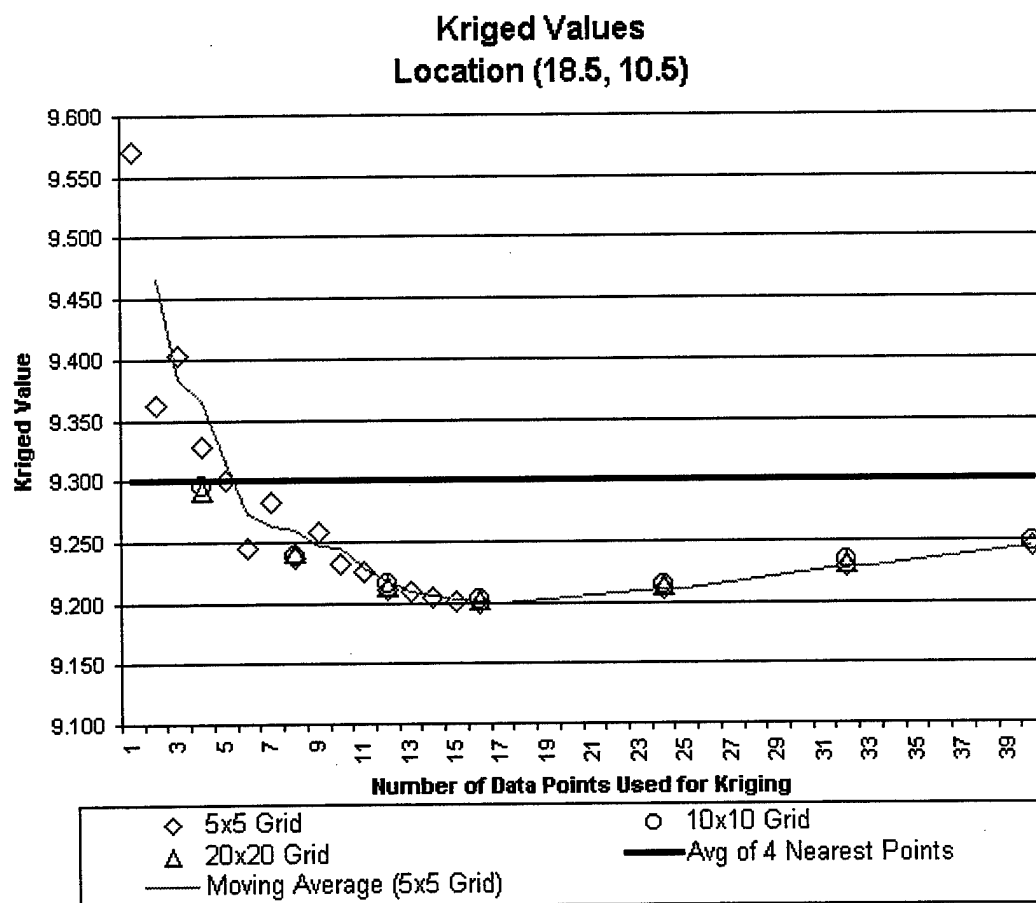


Figure 10 Kriging Values Obtained at Location (18.5, 10.5)

Using the same location, what advantage does one grid size have over another?

Figure 11 shows the execution times of three different grid sizes. It is apparent that if

you double the size of the grid, you more than double the execution time. If a 20 x 20 grid is used to obtain a sample estimate from 16 data points, then the user would have to wait about 7.5 seconds for each sample. This means that if the user takes 15 samples from the site, the user would have to wait almost two minutes to get the results back. For a 10 x 10 grid, this is reduced to 24 seconds, and for a 5 x 5 grid, this is almost six seconds. Therefore, for subsequent simulations, a grid size of 5 x 5 will be used.

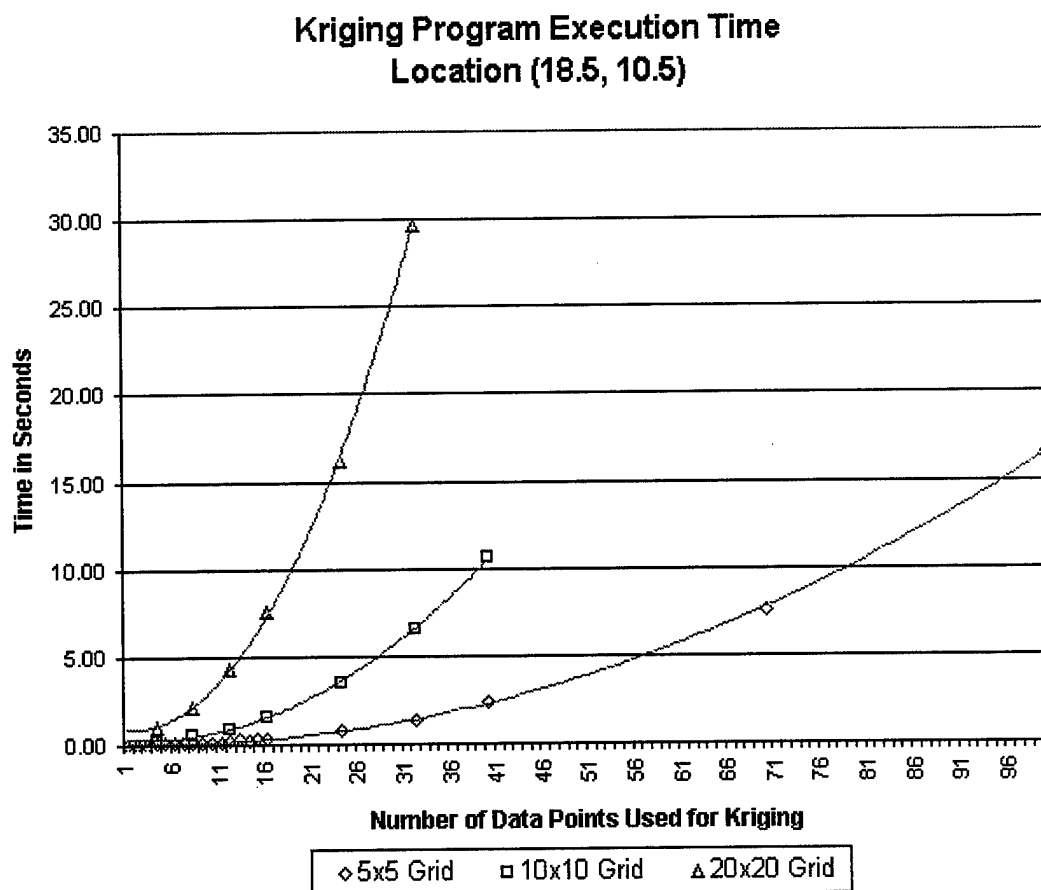


Figure 11 Execution Times of Different Grid Sizes

The next location used was (22.0, 12.0). This location was close to the edge of the data and the data values were fairly high. The purpose of this simulation was to evaluate the performance of the kriging algorithm when the location is next to the edge of the data as well as in a spatially variable area. This time, the location is at a data point;

therefore, the value of this location is indicated by a dark line in Figure 12. However, this figure is misleading because the average of the nine surrounding data points is 15.468, which actually falls below the kriged values in Figure 12. Again, if the first value is dropped (because it only uses one data point), the values range from 16.455 to 15.757. This amounts to about a 4.4 percent difference in values

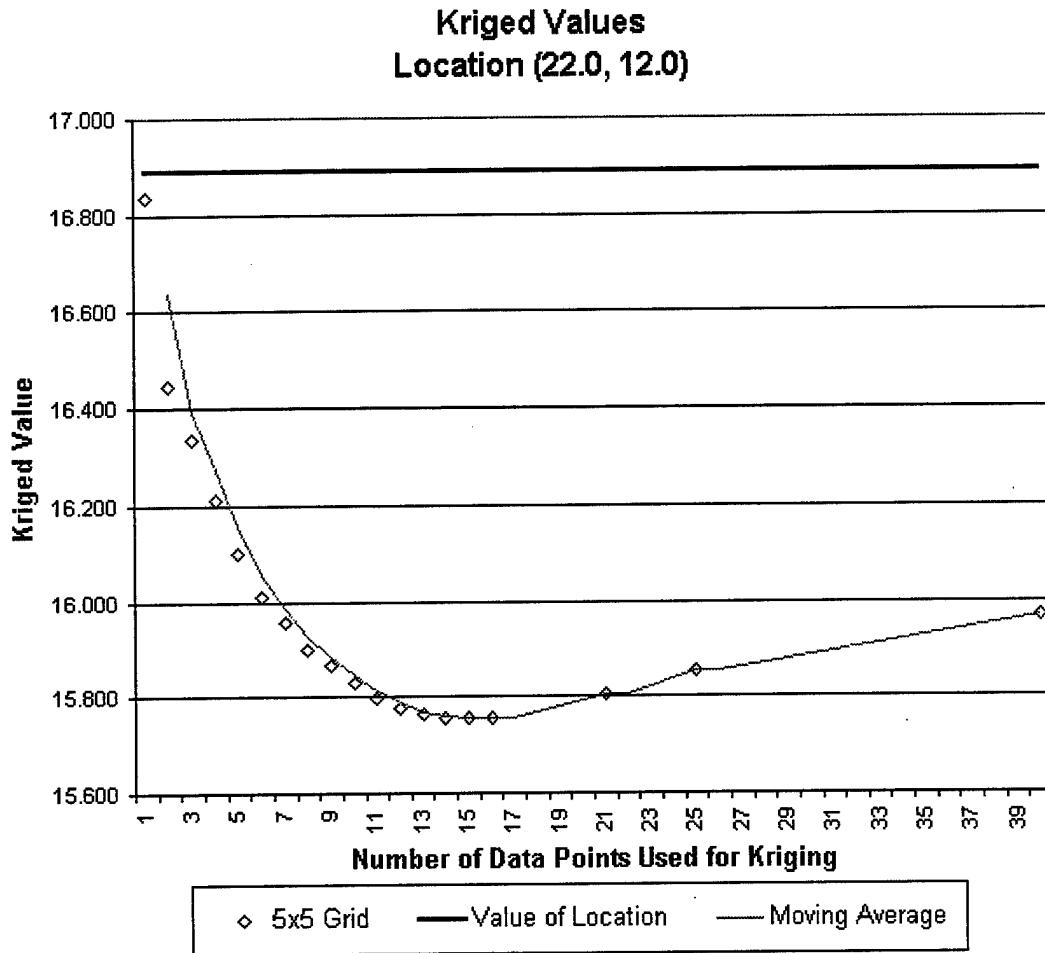


Figure 12 Kriging Values Obtained at Location (22.0, 12.0)

Figure 13 shows the values obtained from (14.2, 16.6), which was the last location used. This location is in a low-value area; therefore, the values are not expected to decrease like those of the previous two simulations. The purpose of this simulation was to evaluate the performance of the kriging algorithm when the location is in a less

variable area and the surrounding data points are increasing. Again, if the first two values are dropped, the values range from .622 to .683. This amounts to about a 9.8 percent difference in values.

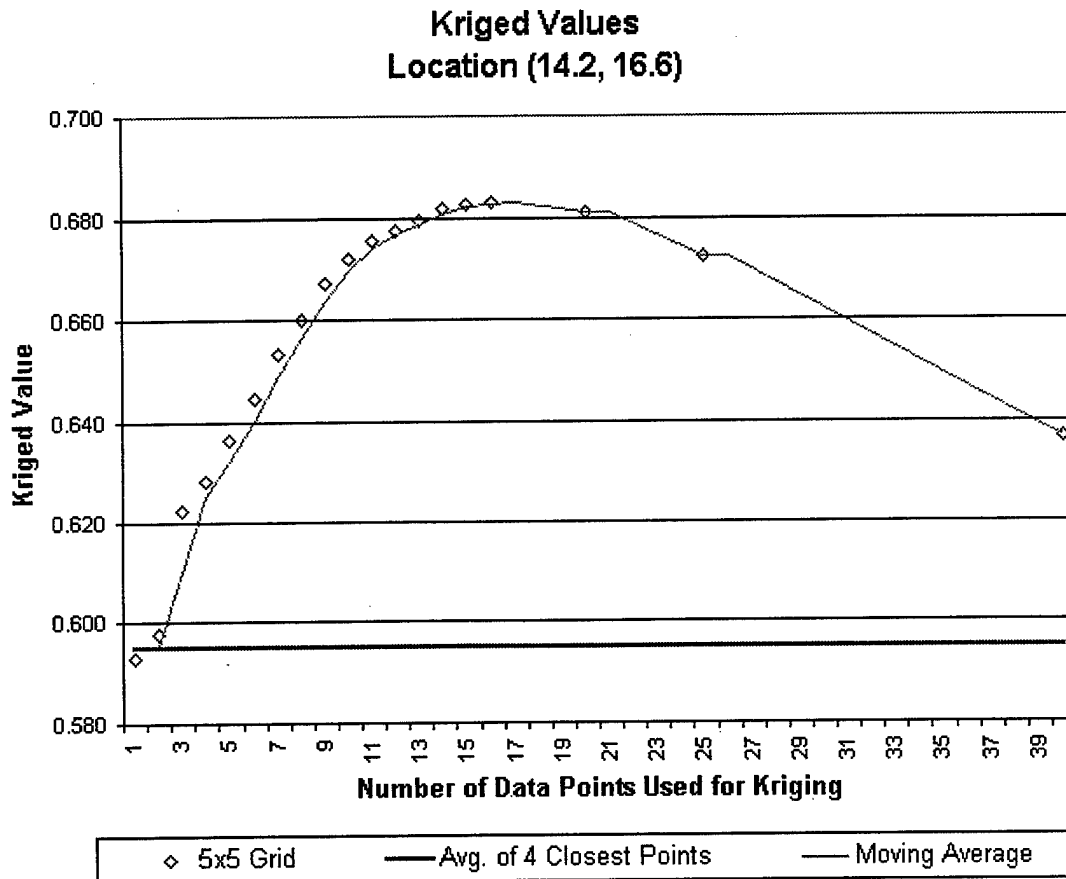


Figure 13 Kriging Values Obtained at Location (14.2, 16.6)

All of the simulations thus far have been using a variogram model function called the spherical model. However, there are three other models that kriging can use, namely, the exponential, Gaussian, and power models. Each of the functions is characterized by a **nugget effect**, contribution, and range (Figure 14). The nugget represents a minimum variance. The contribution is sometimes called the "sill" and represents the average variance of points at such a distance away from the point in question that there is no

correlation between the points. The range represents the distance at which there is no longer a correlation between the points.

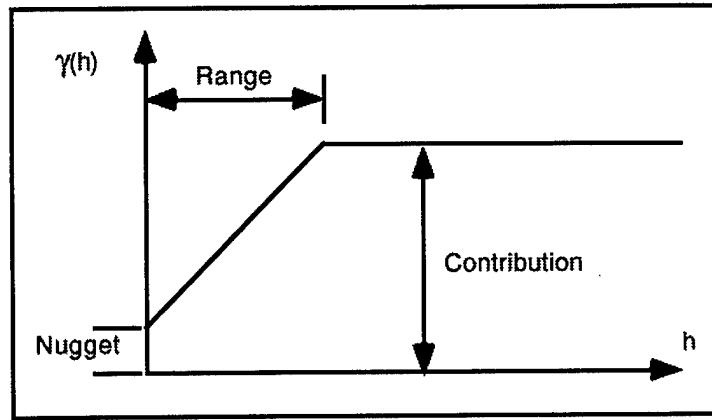


Figure 14 The Parameters Used to Define a Model Variogram

The four variogram model functions are as follows:

- Spherical (range a and contribution c)

$$\gamma(h) = c \left[1.5 \frac{h}{a} - 0.5 \left(\frac{h}{a} \right)^3 \right], \text{ if } \frac{h}{a} \leq a$$

$$\gamma(h) = c, \text{ if } \frac{h}{a} > a$$

Equation 4

- Exponential (parameter a and contribution c)

$$\gamma(h) = c \left[1 - \exp \left(-\frac{3h}{a} \right) \right]$$

Equation 5

- Gaussian (parameter a and contribution c)

$$\gamma(h) = c \left[1 - \exp \left(-\frac{3h^2}{a^2} \right) \right]$$

Equation 6

- Power (power $0 < a < 2$ and slope c)

$$\gamma(h) = c \cdot h^a$$

Equation 7

How do these model functions compare with one another? Figure 15 shows another simulation done at location (18.5, 10.5). It appears that the spherical and exponential models are much more robust for this data set than the other two, with the spherical model behaving slightly more consistent. However, the choice of variogram parameters was arbitrary and by no means optimal. The effects of arbitrarily changing variogram parameters will be discussed later (see Figure 17 and Figure 18)

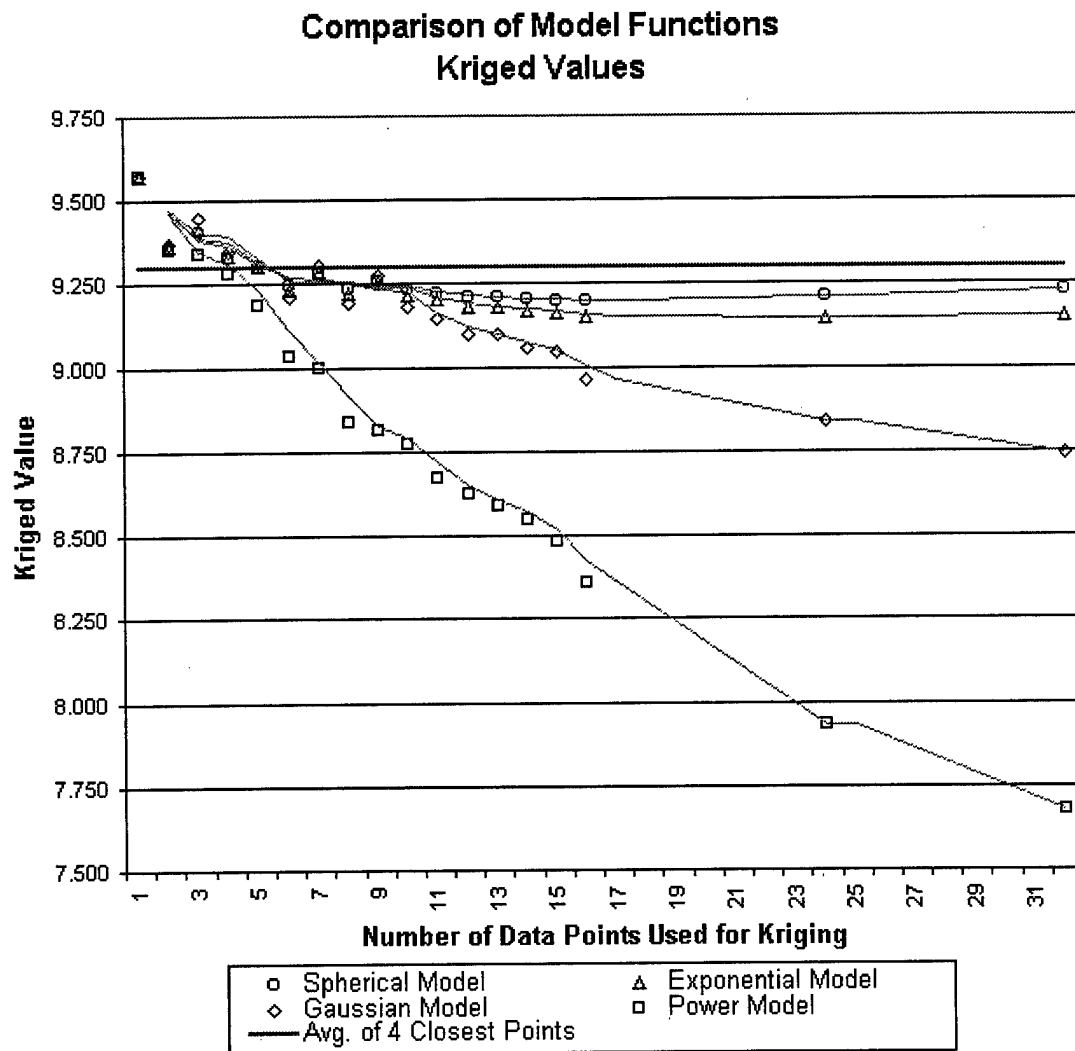


Figure 15 Comparison of Kriging Model Functions

Figure 16 shows the execution times for the four variogram model functions.

From this graph, it is obvious that choosing one model function over another makes no difference in execution time.

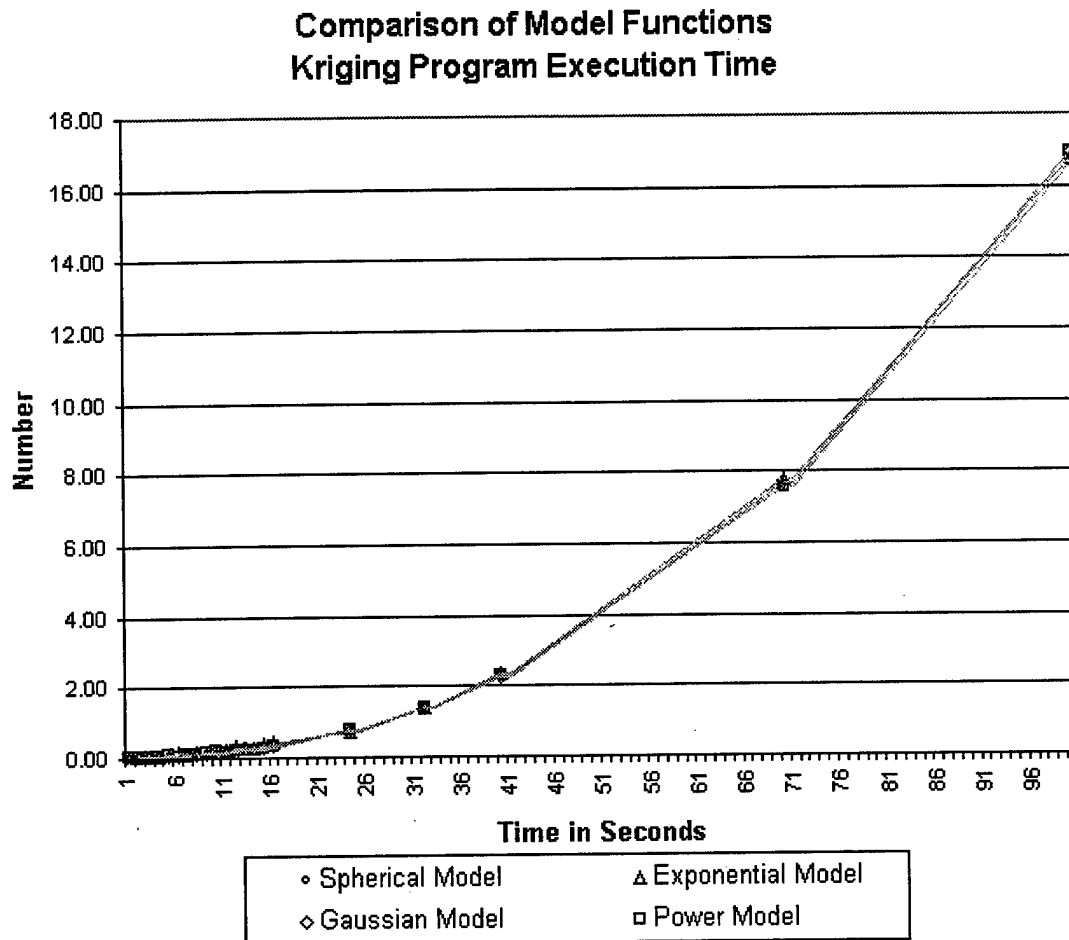


Figure 16 Execution Time for Kriging Model Functions

To further compare the spherical and exponential models, some of the kriging variogram parameters now need to be changed. According to Deutsch and Journel (1992), an acceptable semivariogram model consists of an isotropic nugget effect and any positive linear combination of the standard semivariogram models. The variogram models are influenced by parameters a and c , which may be different, in both meaning and value, for each model. Therefore, these values are the ones to be changed to

determine the effects of changing parameters. Figure 17 shows the results of three different parameter settings on the spherical model. The parameters in the legend are: *nst* for the number of structures, *c0* for the isotropic nugget effect, *cc* for the *c* parameter, *ang* for the azimuth angle, *aa* for the *a* parameter, and *anis* for the anisotropy factor. A detailed description of these parameters can be found in Deutsch and Journel (1992).

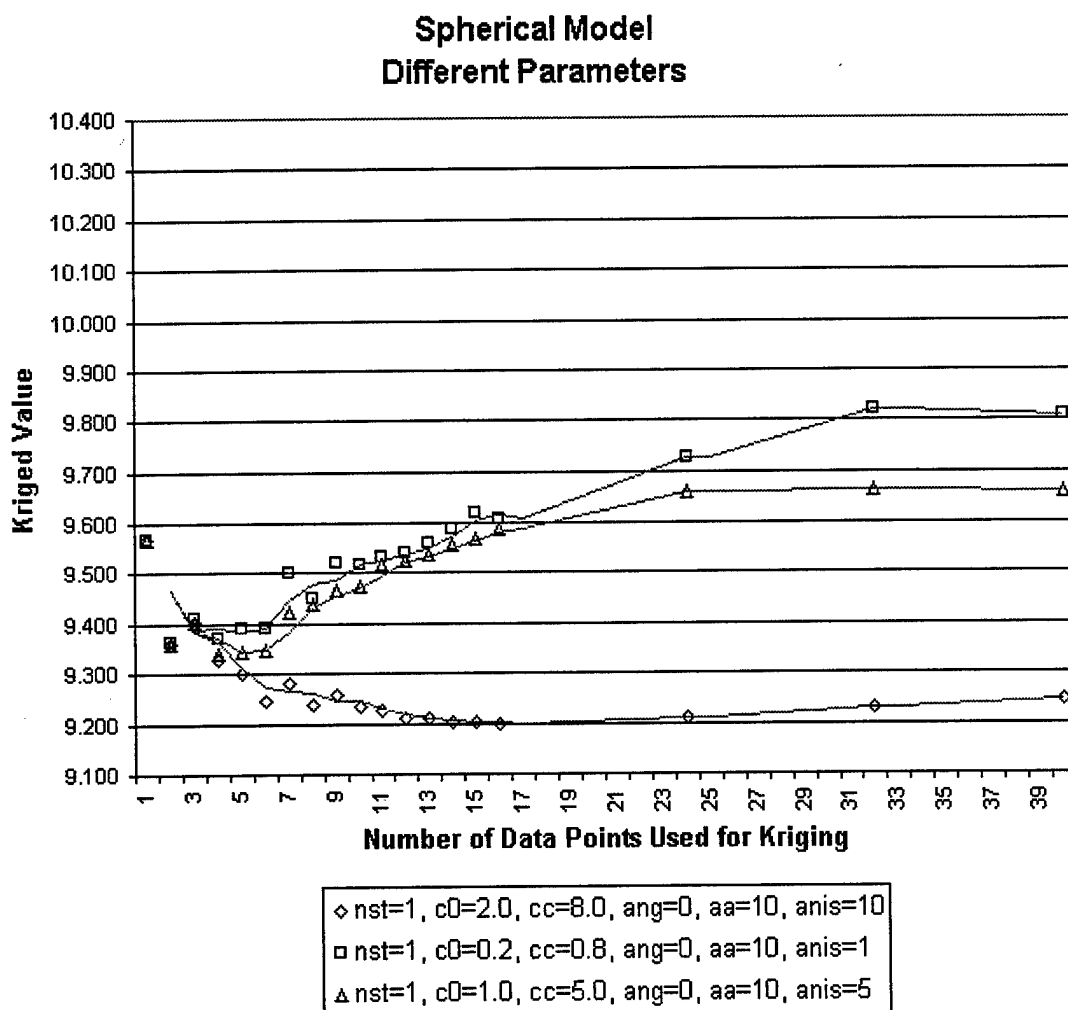


Figure 17 Effects of Parameter Changes on the Spherical Model

Figure 18 shows the results of three different parameter settings on the exponential model. The effects of changing variogram parameters on the exponential

model is more defined when the nugget effect, c parameter, and anisotropy factor are low. This is mainly due to the way each model uses the various parameters.

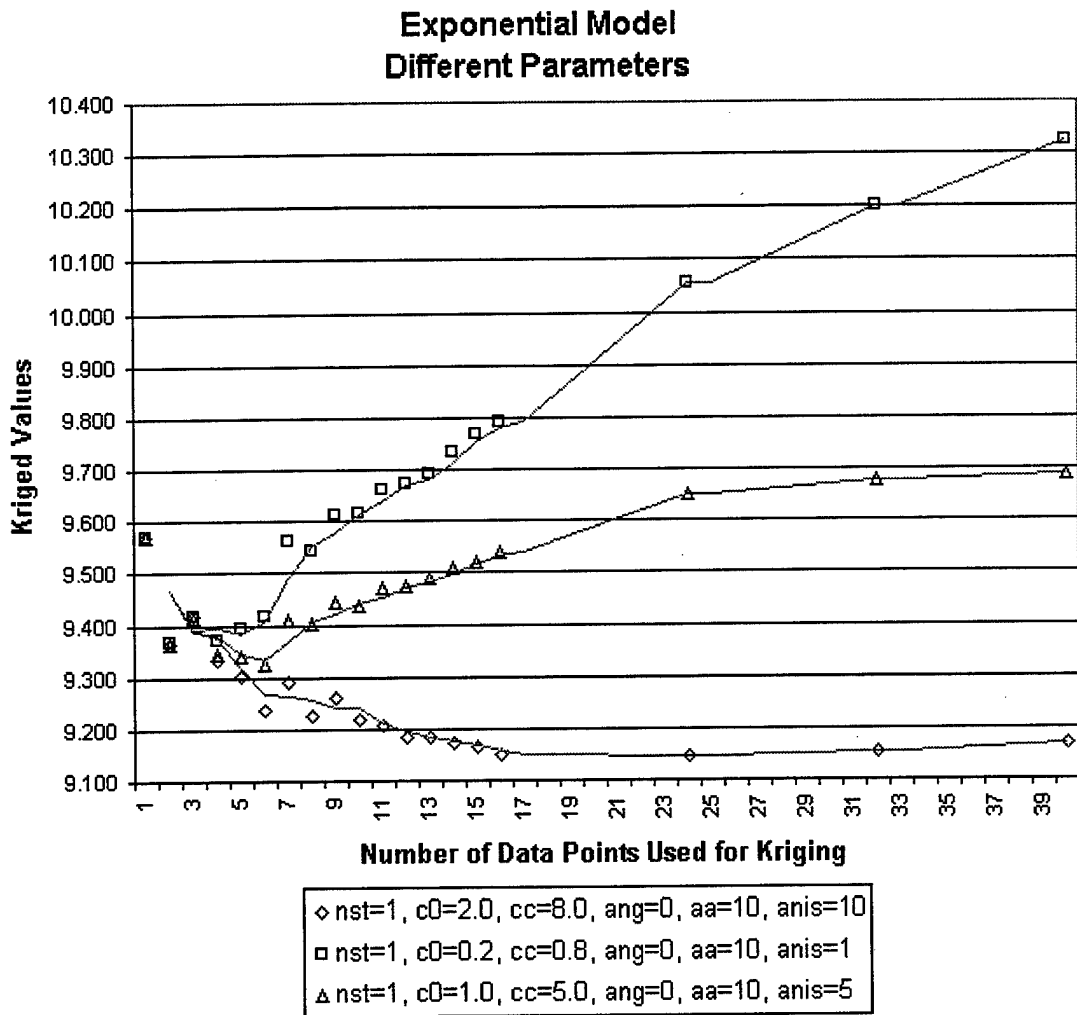


Figure 18 Effects of Parameter Changes on the Exponential Model

The previous two figures showed the effects of arbitrarily changing variogram parameters. However, is there a better method available to help choose these parameters, and which model function is the best to use? GMS provides a graphical interface, called the *Variogram Editor* (Brigham Young University, Feb. 17, 1998), in order to simplify the variogram computation process. The *Variogram Editor* is activated in GMS by selecting the *Edit Variogram* button in the *Kriging Options* dialog. The editor plots the

experimental variogram and the model variogram and allows the user define the model variogram parameters. A model variogram is designed to fit the experimental variogram.

In order to use the *Variogram Editor* in GMS, a scatter point file is required. Therefore, a tabular scatter point file was created from the 50 x 50 grid shown in Figure 8. See Section 4.1.1 for the description of a tabular scatter point file. This scatter point file was then imported into GMS and the *Variogram Editor* was used to find suitable variogram parameters for each of the model functions. The next four figures show a segment of the graphical representation from the *Variogram Editor* that was used to determine the variogram parameters for all four model functions. The purpose was to fit the model function's curve (the smooth curve) as close as possible to the experimental variogram (the curve connecting the dots).

A model variogram is normally constructed using a combination of one or more model functions (Brigham Young University, Feb. 17, 1998). Each instance of a model function is called a "nested structure". A nested structure is created by selecting the *New* button in the lower right corner of the dialog (not shown in the figures). A new structure is created and added to the list of nested structures. The model variogram plotted in the variogram plot window represents the combination of all of the nested structures in the list. The nugget is the same for all nested structures, only the contribution and range of each structure are summed. However, for simplicity, the model variograms that follow were only constructed using one model function.

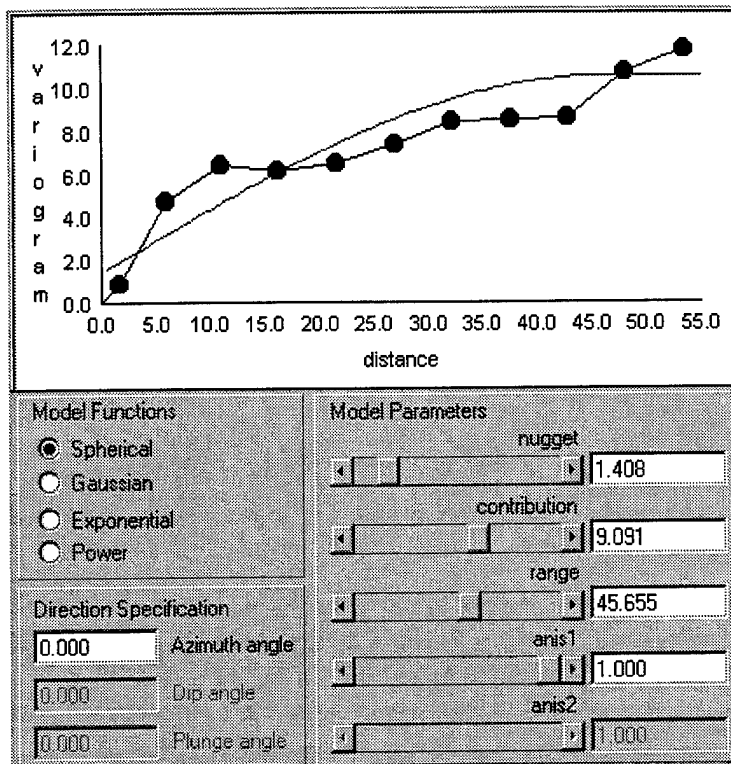


Figure 19 Spherical Model Variogram Fitting

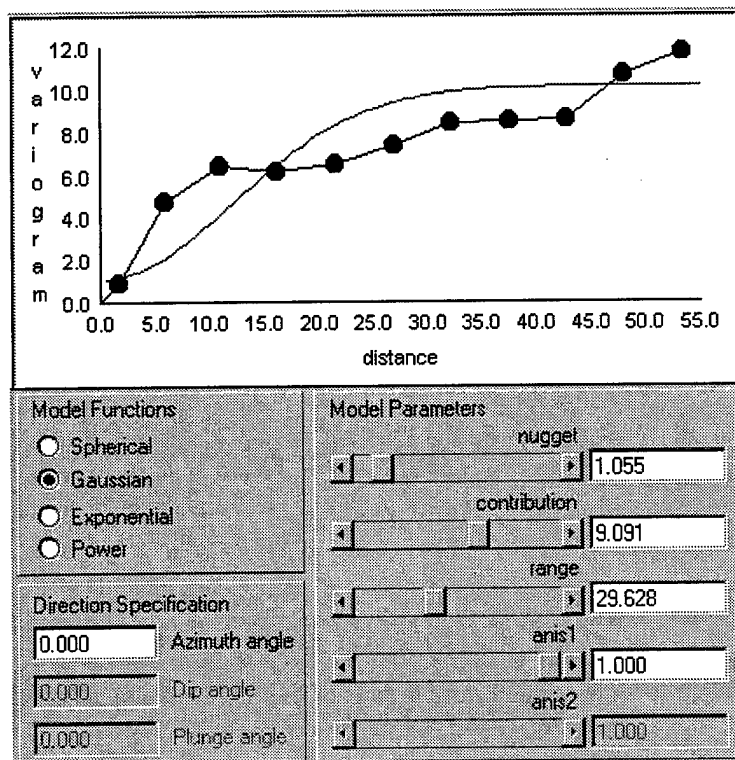


Figure 20 Gaussian Model Variogram Fitting

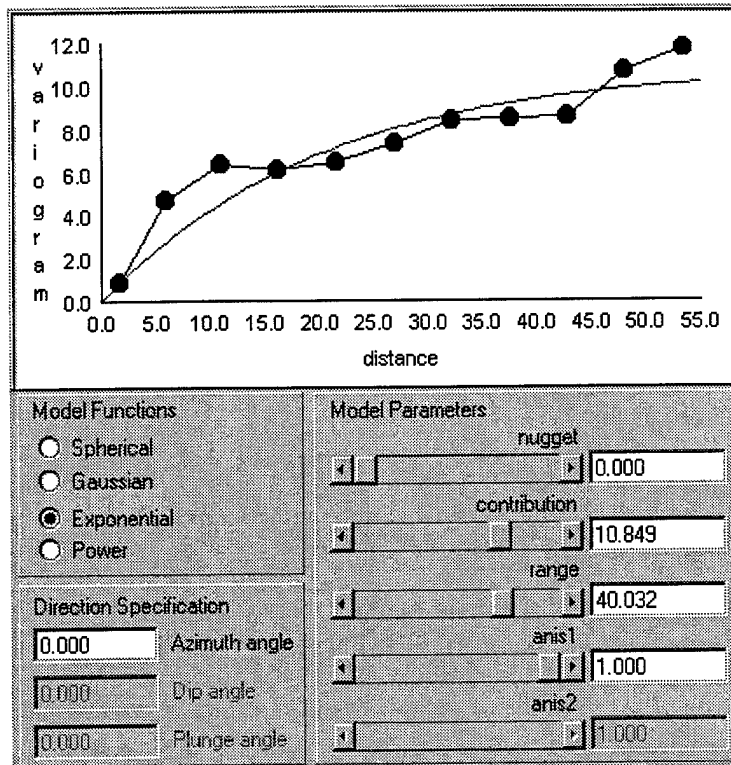


Figure 21 Exponential Model Variogram Fitting

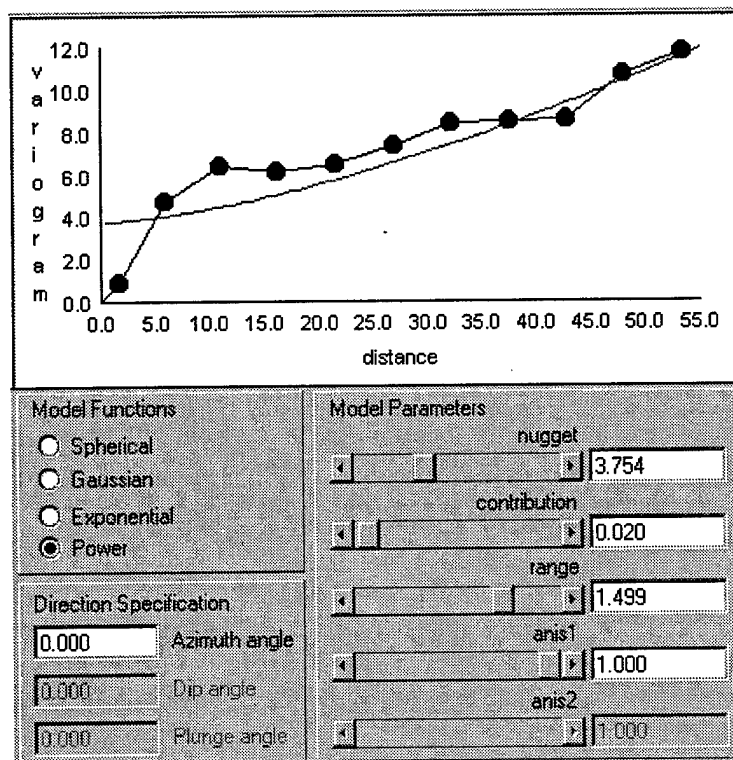


Figure 22 Power Model Variogram Fitting

The variogram parameters determined from the *Variogram Editor* were used on an arbitrary location on the 50 x 50 grid. The results of this variogram fitting are displayed in Figure 23. This time, the power and spherical models appeared to be more consistent with this data set at the location being kriged. This figure also indicates that using just one nested structure may not be the ideal method.

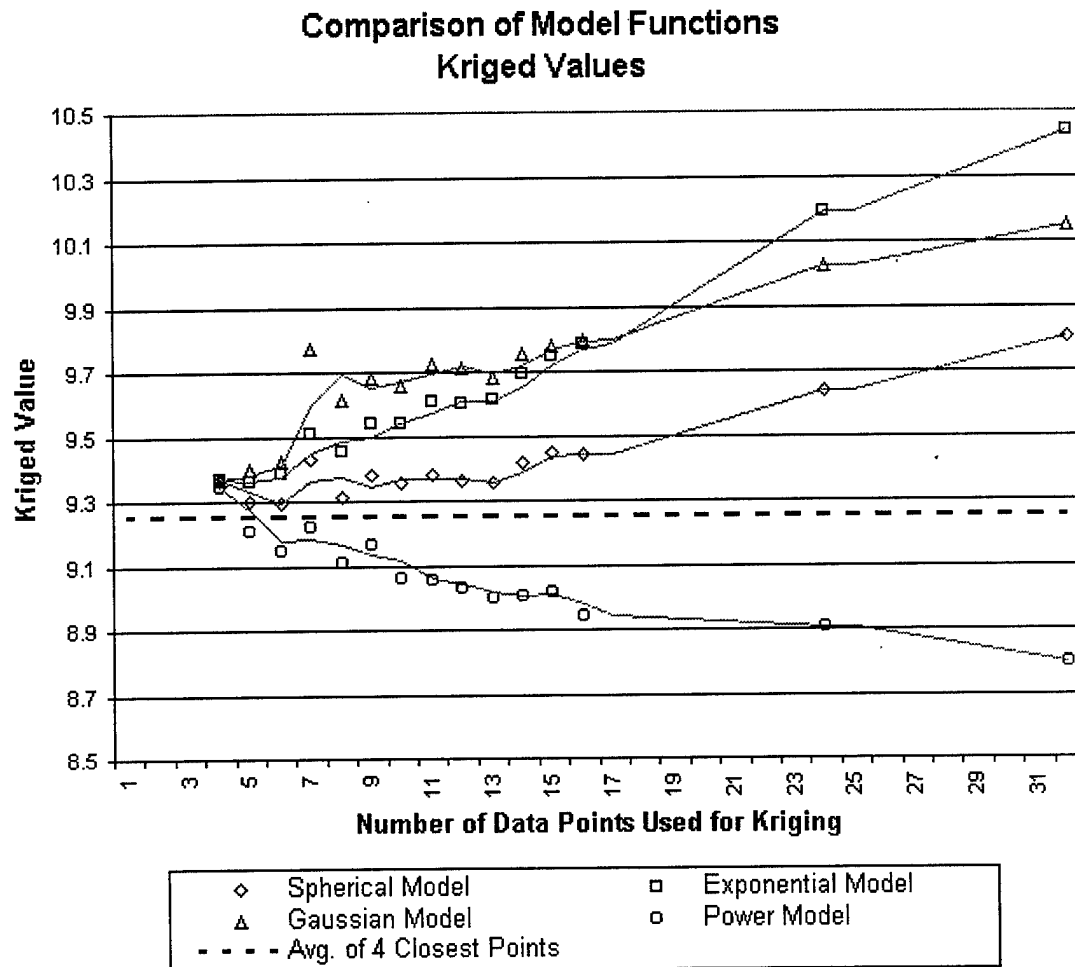


Figure 23 Comparison of Model Functions Based on Variogram Fitting

The last experiment done with the VB6 kriging program was designed to determine the effects of changing the estimation block radius. Figure 24 shows the effects of decreasing the size of the estimation block radius. It is apparent that the values tend to level out at a certain point. This is expected because the smaller the estimation

block radius, the more similar each grid cell should be. In terms of a real-world estimation block radius, if the node spacing of the "real" site is about one meter and the estimation block radius of the sampling method is two centimeters, the corresponding estimation block radius in Figure 24 is 0.02. At that point, the kriged values show little change as the estimation block radius decreases.

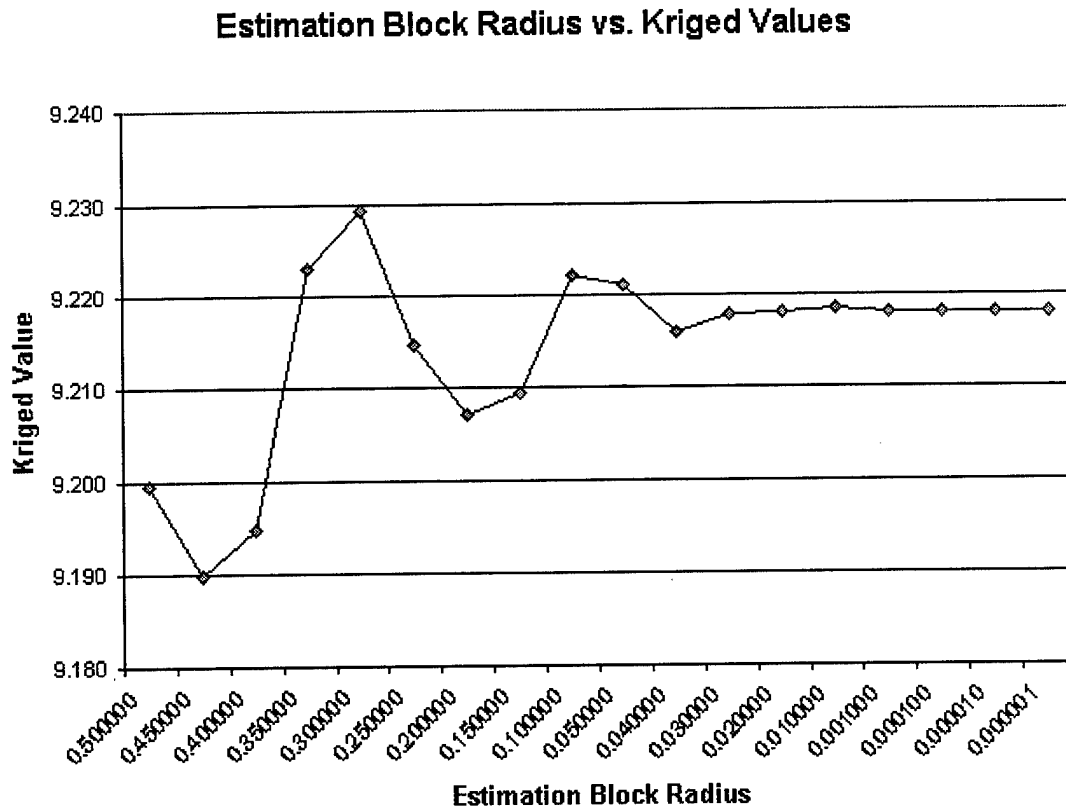


Figure 24 Effects of Decreasing Estimation Block Radius

What happens as the estimation block radius increases? Figure 25 shows the effects of increasing the size of the estimation block radius. This was done to see what happens with the kriging program as the estimation block radius increases beyond the size of the grid. The location used for the center of the estimation block was the center of the 50 x 50 grid. The figure has hills and valleys due to the radius covering certain features in the gridded data. For instance, it is expected that the values will increase

when the estimation block radius reaches around ten units. This is mainly because there is an increase in data values in a few locations starting at about ten units from the center. When the estimation block radius is larger than the grid, the values get erratic because kriging does not consider grid cells that are outside the data range and the values that are considered are obtained from detached locations. That is, as the estimation block radius increases, there is less overlap in the data used to return the kriged value. Consequently, when the estimation block radius is very large, around 200 in this case, kriging returns a value of zero because the centers of all of the grid cells are outside the range of the data. However, if an estimation block covers two or more data points, then the average of these points is returned instead of a kriged value. Therefore, the results of this test are irrelevant to Sim-Site.

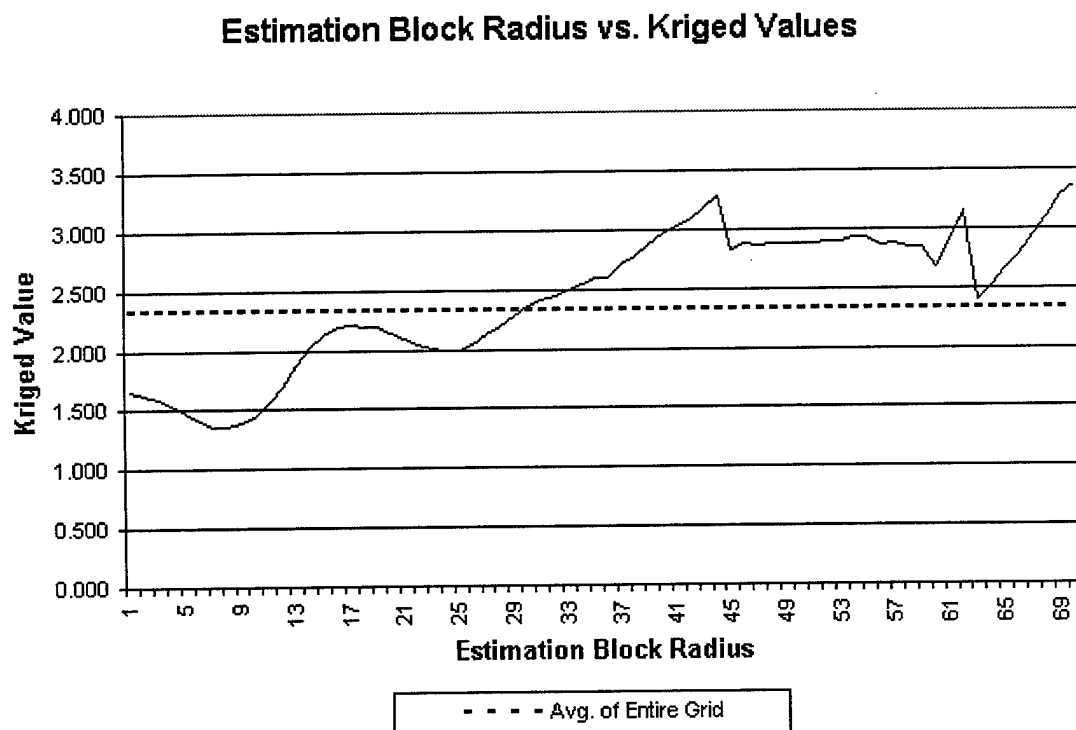


Figure 25 Effects of Increasing Estimation Block Radius

5. Conclusions and Recommendations

5.1 Kriging

5.1.1 Cross Validation

Kriging is a complex linear interpolation method with many different configurations. As was shown in Figure 17 and Figure 18 in Section 4.5, choosing the best variogram parameters may be difficult. In fact, even choosing the correct variogram model can be important. In both the spherical and exponential model, it was shown that choosing different variogram parameters could invert your values. The correct values to choose are based on experience. Deutsch and Journel (1992) pointed out that there are many interdependent subjective decisions in a geostatistical study that it is good practice to validate the entire geostatistical model and kriging plan prior to any production run. They also mentioned that it is not considered good practice to arbitrarily change the variogram parameters to achieve a good cross-validation. This "arbitrary" changing of variogram parameters was done throughout the simulations in section 4.5.

To help the scenario builder in Sim-Site, as well as the software engineer, develop feasible variogram models and search strategies, cross-validation is recommended. Deutsch and Journel (1992) discuss cross-validation and also provide the FORTRAN code to cross-validate a kriging plan. If this cross-validation can be automated for Sim-Site during scenario development, undesired results could be minimized.

5.1.2 GMS Variogram Editor

GMS performs ordinary and universal kriging on scatter point data sets. Therefore, another alternative to cross-validation is to create the "real" site using scatter point data files. Then the *GMS Variogram Editor* can be used to select the nugget effect,

the c parameter, the a parameter, the anisotropy, and the type of model function. More information on the *Variogram Editor* can be found in the *GMS v2.1 Reference Manual* (Brigham Young University, Feb. 17, 1998)

Another option that would be better suited to Sim-Site is to create a scatter point file from the "real" site and use the *Variogram Editor* to determine the model variogram. Variograms are associated with steady state data sets or individual time steps of transient data sets. Once a variogram or set of variograms is defined, the variograms are automatically saved inside a data set file when the data set is saved to disk. Sim-Site also has different time steps and uses each season's data to determine the values of samples through kriging and to compare the user's mathematical model with that of the "real" site. Therefore, extracting a model variogram from a GMS data set file would be the logical approach.

5.1.3 *Uncertainty Due to Kriging and Estimation Block Size*

Although kriging is considered optimal, it still has some uncertainty associated with it, which may be related to the estimation block size. However, Sim-Site does not currently factor in this uncertainty when returning a sample value to the user. That is, Sim-Site only returns the kriged value with sample method uncertainty factored in. Therefore, the uncertainty due to kriging and sample estimation block size needs to be determined and factored in.

5.2 *Sim-Site*

5.2.1 *Dimensions*

Currently, Sim-Site is only two-dimensional; therefore, more work is needed to expand it to three dimensions. This will involve implementing three-dimensional

kriging, or other interpolation scheme, along with other changes necessary to add a third dimension.

5.2.2 *Sampling Parameters*

Sim-Site currently only returns hydraulic conductivity; therefore, more sampling parameters, along with their associated sampling methods, need to be added. Many of the sampling parameters are already contained in Sim-Site's menu; however, there is no code to process them. Additionally, the current code for hydraulic conductivity is required to have an exact spelling, which offers no naming flexibility to the scenario builder. That is, a check for "Hydraulic Conductivity" is hard coded in Sim-Site. The parameter names are read in from a data file that the scenario builder can modify. Consequently, if the scenario builder chooses to change the name of a parameter, Sim-Site will not recognize it.

5.2.3 *Scenario Builder*

The person who builds a scenario for Sim-Site must do it by modifying certain data files that specify items like kriging parameters and file names. Another option is to develop a separate program, possibly called the Sim-Site Scenario Builder, to help create a scenario. This program would provide a graphical interface to help the scenario builder easily develop a scenario without having to know about the underlying structure of the data files.

5.2.4 *TIFF Image Processing*

As mentioned in Section 3.3, Sim-Site has to use the Wang *Image Edit* control to process TIFF images used by GMS. A routine was originally developed for Sim-Site to read the TIFF image and draw it directly to a VB6 *PictureBox* control. However, this

method was much too slow and had to be abandoned. This routine can be enhanced to either save the image to the Windows clipboard or to a bitmap file. This would eliminate the need for the *Image Edit* control, the VB6 *PictureClip* control, and the installation of Imaging for Windows.

5.2.5 *General Enhancements*

Sim-Site was designed to resemble most of the common Windows 95 based programs. It has a tool bar, a status bar, a help file, a menu, etc. However, some common functions, like context sensitive help and having a menu appear when the user right clicks on an object, are not implemented in Sim-Site. These features were considered as "nice to have" and can be easily added if desired.

Appendix A: Site Characterization Simulator Specification

1. Introduction

Figure 26 contains the object diagram for Sim-Site. Each of the objects is explained in Section 2. This figure represents the current configuration of Sim-Site; however, the specification is subject to change as new requirements and features are added.

For each object, the file name of the object will be listed first followed by a brief description of the object. Files are either modules (*.bas), classes (*.cls), or forms (*.frm). Then the properties (also known as attributes) will be listed. Visual Basic provides three kinds of property procedures, as described in the following table.

Table 3 Visual Basic Property Procedures

Procedure	Purpose
Property Get	Returns the value of a property.
Property Let	Sets the value of a property.
Property Set	Sets the value of an object property (that is, a property that contains a reference to an object).

Each of these property procedures has a particular role to play in defining a property.

The typical property will be made up of a pair of property procedures: A *Property Get* to retrieve the property value, and a *Property Let* or *Property Set* to assign a new value.

The property procedures that apply to each property will be identified along with its data type. Finally, the object's public and private methods will be listed with a brief description of each method. More in-depth descriptions of properties and methods can be found in the code comments.

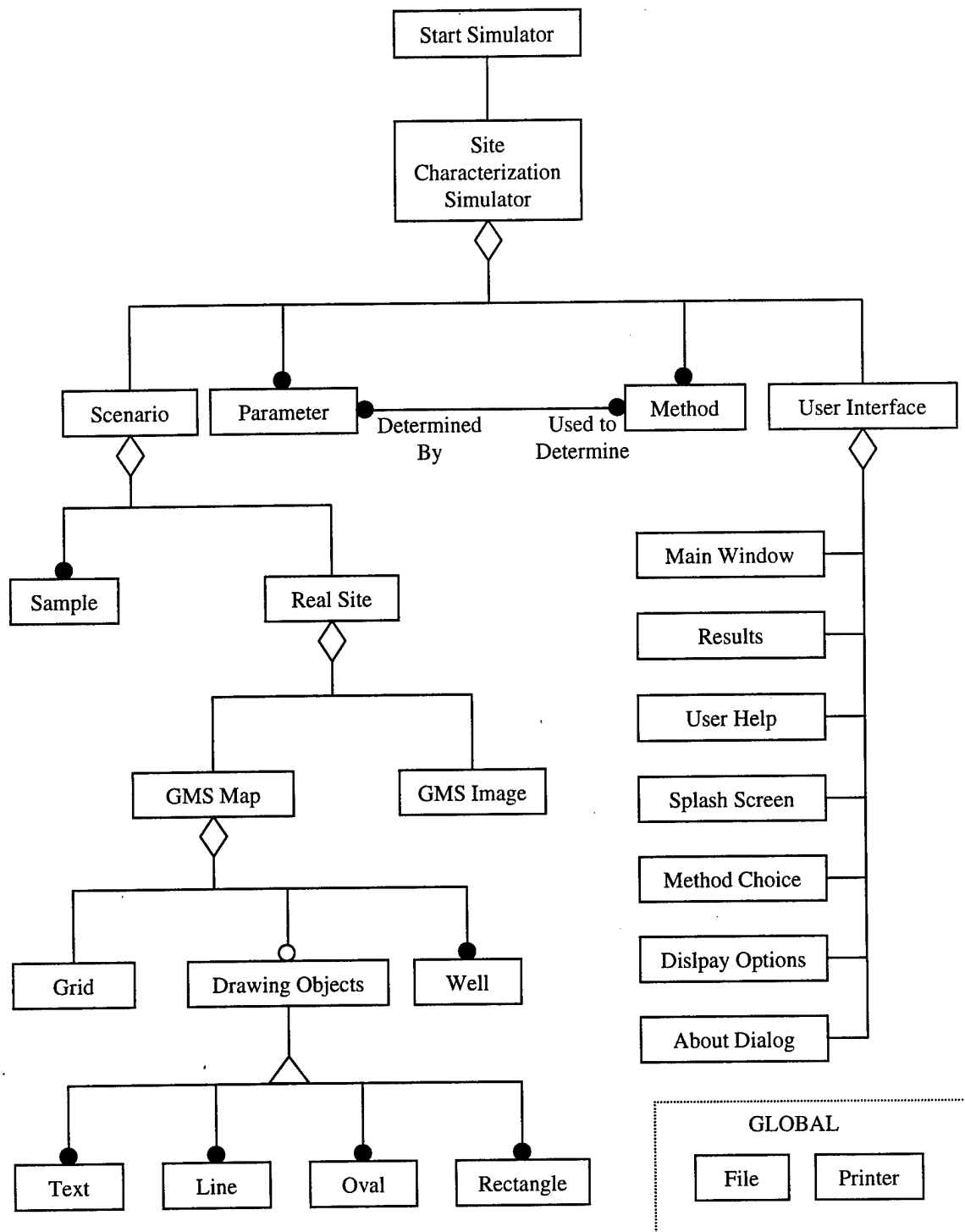


Figure 26 Sim-Site Object Diagram

2. Objects

Two events are present in every object (forms and classes); therefore, they will not be listed for each object. These procedures only respond to *Initialize* and *Terminate* events and cannot be called by other procedures:

Private Sub *object_Initialize*()

Occurs when an instance of an object is created.

Private Sub *object_Terminate*()

Occurs when all references to an instance of an object are removed from memory by setting all the variables that refer to the object to *Nothing* or when the last reference to the object falls out of scope.

However, the *Initialize* event has no context (Swartzfager et al., 1999). It does not know what created the class or why it was created. No information can be passed to the class or event to let it know how to initialize. Swartzfager et al. (1999) suggested that a better way to initialize a class is to provide it with an initialization method. Therefore, each class and form has its own initialization method called *Init_Class* along with a private variable called *mblnInitialized*. This method and variable are not listed individually in the object descriptions that follow. That is, if "None" is listed under public methods for an object, then there are no other public methods besides *Init_Class*.

Sample, *Text*, *Line*, *Oval*, *Rectangle*, *Well*, *Method*, and *Parameter* classes can all have zero or more instances at any given time. Therefore, these classes also have a corresponding collection class to hold all of the instances of the class. All of these collection classes have the same methods and only one property. In all cases, the

property is a descriptive variable of type *Collection*. For instance, there is a *SampleSet* class with the variable declaration:

Private mcolSamples As New Collection

The methods that all of the collection classes have in common are:

Public Function Add() As String

This returns a key value (similar to a pointer) of the instance just added.

Public Property Get Item(ByVal Index As Variant) As *object*

Used when referencing an element in the collection, where *object* is the class that applies to the collection. This should be the default procedure for the class.

Syntax: Set *obj* = *col*.Item("Sample 5") or Set *obj* = *col*.Item(5).

Public Property Get Count() As Long

Used when retrieving the number of elements in the collection.

Public Sub Remove(ByVal Index As Variant)

Used when removing an element from the collection.

Public Property Get NewEnum() As IUnknown

This property allows you to enumerate the collection with the *For...Each* syntax. This procedure should have already been marked as hidden with a Procedure ID of -4 via the *Procedure Attributes* dialog box, so that the collection will work with *For...Each*.

Only the file names of the collection classes, not the collection classes themselves, will be shown in the description of the objects that follow.

2.1 *Start Simulator*

File Name: SimStart.bas

Description: This is the startup object for Sim-Site.

2.1.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
Simulator	Object	X		X

2.1.2 *Public Methods*

Sub Main()

This is the startup procedure for Sim-Site.

2.1.3 *Private Methods*

None.

2.2 *Site Characterization Simulator*

File Name: Simulator.cls

Description: This object contains the scenario, sample parameters, sample methods, and user interface objects.

2.2.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
Scenario	Object	X		X
UserInterface	Object	X		X
UserFilename	String	X	X	
Methods	Collection	X		
Parameters	Collection	X		

2.2.2 *Public Methods*

Sub OpenScenario()

Opens a scenario.

Sub OpenUserFile()

Opens the user's file.

Sub RemoveScenario()

Removes the current scenario.

Sub SaveUserFile()

Saves the user's file.

2.2.3 Private Methods

Sub GetSampleMethods()

Retrieves the sample methods from a data file.

Sub GetSampleParameters()

Retrieves the sample parameters from a data file.

2.3 Scenario

File Name: Scenario.cls

Description: This object represents the scenario. It contains the user's samples and the real site.

2.3.1 Properties

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
Budget()	Double	X	X	
BudgetLeft()	Double	X	X	
CurrentSeason	Integer	X	X	
Filename	String	X	X	
Modified	Boolean	X	X	
NumSeasons	Integer	X	X	
PictureIndex	Integer	X	X	
RealSite	Object	X		X
SeasonLength	Double	X	X	
SampleDepth	Double	X	X	
Samples	Collection	X		
CoreCost()	Double	Private		
CoreType()	String	Private		

CoreUse()	Boolean	Private
WellCost()	Double	Private
WellType()	String	Private
WellUse()	Boolean	Private

2.3.2 *Public Methods*

Sub AddCoreType()

Adds a core type to the *CoreType* array based on the current sample method selected.

Sub AddWellType()

Adds a well type to the *WellType* array based on the current sample method selected.

Sub InitMethodChoices()

Initializes the *CoreType* and *WellType* arrays.

Sub MoveSample()

Moves a sample to a different location.

Sub RemoveSample()

Removes a sample.

Sub UpdateCost()

Updates the cost when the user chooses a different method.

Sub UpdateMethodChoices()

Updates the choices of core or well types along with the cost when a user changes the sample method.

Sub UpdateTakingSampleStatus()

Changes the state depending on where the user is during sampling. That is, selecting a method, selecting a location, modifying a sample, or displaying a sample.

2.3.3 Private Methods

None.

2.4 Sample

File Name: Sample.cls

Collection Class Name: SampleSet.cls

Description: This object represents one user sample and the collection holds each instance of a user sample.

2.4.1 Properties

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
Cost	Double	X	X	
GotResults	Boolean	X	X	
MethodKey	String	X	X	
ParameterKey	String	X	X	
PictureIndex	Integer	X	X	
SeasonNumber	Integer	X	X	
XPosition	Double	X	X	
YPosition	Double	X	X	

2.4.2 Public Methods

Function CheckForSave() As Integer

Prompts the user to save the work if something was changed.

Sub ReadFromFile()

Reads the sample from a user's file.

Sub SaveToFile()

Saves the sample to a user's file.

2.4.3 Private Methods

None.

2.5 Real Site

File Name: RealSite.cls

Description: This object represents the real site to be sampled. It contains a GMS image of the site and a corresponding GMS map representation.

2.5.1 Properties

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
GMSImage	Object	X		X
GMSMap	Object	X		X
ImageFilename	String	X	X	
MapFilename	String	X	X	

2.5.2 Public Methods

Sub GetRealSite()

Loads the GMS image and the GMS map.

2.5.3 Private Methods

None.

2.6 GMS Map

File Name: GMSMap.cls

Description: This object represents the GMS map imported from GMS. It contains a grid, some drawing objects, and some wells.

2.6.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
DrawingObjects	Object	X		X
Filename	String	X	X	
Grid	Object	X		X
Wells	Collection	X		
LineStatuses	Enum	Private		

2.6.2 *Public Methods*

Sub OpenMap()

Opens, reads, and displays the GMS map file.

Sub Draw ()

Polymorphic procedure to draw a drawing object, well, or grid frame on the site.

2.6.3 *Private Methods*

None.

2.7 *GMS Image*

File Name: GMSImage.cls

Description: This object represents the GMS image imported from GMS.

2.7.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
ClipPointXMin	Double	X	X	
ClipPointXMax	Double	X	X	
ClipPointYMin	Double	X	X	
ClipPointYMax	Double	X	X	
Filename	String	X	X	
ImageToWorldX	Double	X		
ImageToWorldY	Double	X		
RegPointImageX(3)	Double	X	X	
RegPointImageY(3)	Double	X	X	
RegPointWorldX(3)	Double	X	X	
RegPointWorldY(3)	Double	X	X	
TIFFFile	String	X	X	

2.7.2 *Public Methods*

Public Sub OpenImage()

Opens, reads, and displays the GMS image.

2.7.3 *Private Methods*

None.

2.8 *Grid*

File Name: Grid.cls

Description: This object represents the grid and its attributes imported from GMS and MODFLOW.

2.8.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
GridColSpace()	Double	X	X	
GRIDFRAMEdx	Double	X	X	
GRIDFRAMEdy	Double	X	X	
GRIDFRAMEdz	Double	X	X	
GRIDFRAMErotate	Double	X	X	
GRIDFRAMEx	Double	X	X	
GRIDFRAMEy	Double	X	X	
GRIDFRAMEz	Double	X	X	
GridRowSpace()	Double	X	X	
HydrCond()	Double	X	X	
NumColumns	Integer	X	X	
NumRows	Integer	X	X	

2.8.2 *Public Methods*

None.

2.8.3 *Private Methods*

None.

2.9 *Drawing Objects*

File Name: DrawingObjects.cls

Description: This object contains the four drawing objects from GMS: text, lines, ovals, and rectangles.

2.9.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
Lines	Collection	X		
Rectangles	Collection	X		
Ovals	Collection	X		
DisplayText	Collection	X		

2.9.2 *Public Methods*

None.

2.9.3 *Private Methods*

None.

2.10 *Well*

File Name: Well.cls

Collection Class Name: WellSet.cls

Description: This object represents one well and the collection holds each instance of a well.

2.10.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
ID	Integer	X	X	
SPECHHEADconstflag	Integer	X	X	
SPECHHEADvalue	Double	X	X	
StatusBarText	String	X	X	
WELLCONCconstflag	Integer	X	X	
WELLCONCvalue	Double	X	X	
WELLFLUXconstflag	Integer	X	X	
WELLFLUXvalue	Double	X	X	
XPosition	Double	X	X	
YPosition	Double	X	X	
ZPosition	Double	X	X	

2.10.2 *Public Methods*

None.

2.10.3 *Private Methods*

None.

2.11 *Text*

File Name: DisplayText.cls

Collection Class Name: DisplayTextSet.cls

Description: This object represents one text object and the collection holds each instance of a text object.

2.11.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
COLORb	Integer	X	X	
COLORg	Integer	X	X	
COLORr	Integer	X	X	
FILLb	Integer	X	X	
FILLg	Integer	X	X	
FILLr	Integer	X	X	
LOCALrotate	Double	X	X	
LOCALx	Double	X	X	
LOCALy	Double	X	X	
PCFONTbold	Boolean	X	X	
PCFONTitalic	Boolean	X	X	
PCFONTname	String	X	X	
PCFONTsize	Integer	X	X	
TheSTRING	String	X	X	
UNIXFONT	Integer	X	X	

2.11.2 *Public Methods*

None.

2.11.3 *Private Methods*

None.

2.12 *Line*

File Name: ALine.cls

Collection Class Name: LineSet.cls

Description: This object represents one line object and the collection holds each instance of a line object.

2.12.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
ARRHED	Integer	X	X	
LINECOLb	Integer	X	X	
LINECOLg	Integer	X	X	
LINECOLr	Integer	X	X	
HEDLEN	Integer	X	X	
HEDWID	Integer	X	X	
PTbegx	Double	X	X	
PTbegy	Double	X	X	
PTbegz	Double	X	X	
PTendx	Double	X	X	
PTendy	Double	X	X	
PTendz	Double	X	X	
STYLE	Integer	X	X	
THICK	Integer	X	X	
VERTS	Integer	X	X	

2.12.2 *Public Methods*

None.

2.12.3 *Private Methods*

Function Arcsin() As Double

Returns the arc sine of a number in radians. Only used when drawing arrows.

2.13 *Oval*

File Name: Oval.cls

Collection Class Name: OvalSet.cls

Description: This object represents one oval object and the collection holds each instance of an oval object.

2.13.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
ALPHA	Double	X	X	
C1x	Double	X	X	
C1y	Double	X	X	
C1z	Double	X	X	
C2x	Double	X	X	
C2y	Double	X	X	
C2z	Double	X	X	
C3x	Double	X	X	
C3y	Double	X	X	
C3z	Double	X	X	
C4x	Double	X	X	
C4y	Double	X	X	
C4z	Double	X	X	
FILLCOLb	Integer	X	X	
FILLCOLg	Integer	X	X	
FILLCOLr	Integer	X	X	
FILLPAT	Integer	X	X	
LINECOLb	Integer	X	X	
LINECOLg	Integer	X	X	
LINECOLr	Integer	X	X	
STYLE	Integer	X	X	
THETA	Double	X	X	
THICK	Integer	X	X	

2.13.2 *Public Methods*

None.

2.13.3 *Private Methods*

None.

2.14 *Rectangle*

File Name: Rectangle.cls

Collection Class Name: RectangleSet.cls

Description: This object represents one rectangle object and the collection holds each instance of a rectangle object.

2.14.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
ALPHA	Double	X	X	
C1x	Double	X	X	
C1y	Double	X	X	
C1z	Double	X	X	
C2x	Double	X	X	
C2y	Double	X	X	
C2z	Double	X	X	
C3x	Double	X	X	
C3y	Double	X	X	
C3z	Double	X	X	
C4x	Double	X	X	
C4y	Double	X	X	
C4z	Double	X	X	
FILLCOLb	Integer	X	X	
FILLCOLg	Integer	X	X	
FILLCOLr	Integer	X	X	
FILLPAT	Integer	X	X	
LINECOLb	Integer	X	X	
LINECOLg	Integer	X	X	
LINECOLr	Integer	X	X	
STYLE	Integer	X	X	
THETA	Double	X	X	
THICK	Integer	X	X	

2.14.2 *Public Methods*

None.

2.14.3 *Private Methods*

None.

2.15 *Parameter*

File Name: Parameter.cls

Collection Class Name: ParameterSet.cls

Description: This object represents one sample parameter and the collection holds each instance of a sample parameter.

2.15.1 Properties

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
ParamName	String	X	X	
DeterminedBy()	String	X	X	
NumMethods	Integer	X	X	

2.15.2 Public Methods

None.

2.15.3 Private Methods

None.

2.16 Method

File Name: Method.cls

Collection Class Name: MethodSet.cls

Description: This object represents one sample method and the collection holds each instance of a sample method.

2.16.1 Properties

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
MethodName	String	X	X	
Uncertainty	String	X	X	
EstBlock	String	X	X	
EstCost	String	X	X	
UsedToDetermine	String	X	X	

2.16.2 Public Methods

Function Cost() As Double

Computes the cost of a sample method from its raw cost string. Example

raw cost string: $1250 + [81 * \text{depth} (m)]$

2.16.3 *Private Methods*

None.

2.17 *User Interface*

File Name: UserInterface.cls

Description: This object contains all of the forms and windows that the user sees and interacts with.

2.17.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
About	Form			
DisplayOptions	Form	X		X
MainWindow	Form	X		X
MethodChoices	Form	X		X
Printer	Object	X		X
Results	Form	X		X
Splash	Form			
UserHelp	Object	X		X
DisplayOptionsStatus	Record	Private		

2.17.2 *Public Methods*

Sub CloseForm()

Polymorphic procedure that closes a form or dialog.

Sub ShowForm()

Polymorphic procedure that displays a form or dialog.

2.17.3 *Private Methods*

Sub ApplyDisplayOptions()

Applies the display options that are checked on the *DisplayOptions* form.

Sub RefreshDisplayOptions()

Refreshes the checked items on the *DisplayOptions* form to their original configuration.

Sub UpdateDisplayOptions()

Updates the stored display options to those on the *DisplayOptions* form.

2.18 Main Window

File Name: frmSimSite.frm

Description: This object contains the main window of Sim-Site.

2.18.1 Properties

None.

2.18.2 Public Methods

Sub InitializeToolBar ()

Initializes the tool bar.

Sub AdjustMenu ()

Polymorphic procedure that adjusts a menu.

2.18.3 Private Methods

Sub ResizePictureControls()

Resizes the picture controls on the form after a *Resize* event occurs.

2.18.4 Private Events Used

Sub Form_Load()

Sub Form_QueryUnload()

Sub Form_Resize()

Sub lblText_MouseMove()

Sub mnuAbout_Click()

Sub mnuContents_Click()

Sub mnuDisplayOptions_Click()

Sub mnuExit_Click()
Sub mnuIndex_Click()
Sub mnuNew_Click()
Sub mnuNextSeason_Click()
Sub mnuOpen_Click()
Sub mnuParameter_Click()
Sub mnuPrint_Click()
Sub mnuPrintSetup_Click()
Sub mnuSampleResults_Click()
Sub mnuSave_Click()
Sub mnuSaveAs_Click()
Sub mnuSearch_Click()
Sub mnuSeasonResults_Click()
Sub picSample_DblClick()
Sub picSample_MouseDown()
Sub picSample_MouseMove()
Sub picSample_MouseUp()
Sub picSiteMoving_Change()
Sub picSiteMoving_Click()
Sub picSiteMoving_DragDrop()
Sub picSiteMoving_MouseMove()
Sub picWell_Click()
Sub picWell_DblClick()

Sub picWell_MouseMove()

Sub tlbSimSite_ButtonClick()

vsbSiteImage_Change()

hsbSiteImage_Change()

2.19 Results

File Name: frmResults.frm

Description: This object contains the results of both the samples and the seasons.

2.19.1 Properties

None.

2.19.2 Public Methods

Sub ShowSampleResults()

Calculates and displays the results of each sample.

Sub ShowSeasonResults()

Calculates and displays the season by season results of the entire simulation.

2.19.3 Private Methods

Function Krige() As Double

Returns a sample result via Kriging of a 2-D rectangular grid.

Function cova2() As Double

Returns the covariance between two points (2-D Version).

Sub ksol()

Returns the solution of a system of linear equations.

2.19.4 *Private Events Used*

Sub cmdClose_Click()

Sub cmdPrint_Click()

Sub cmdSaveRTF_Click()

Sub cmdSaveScatter_Click()

Sub Form_Resize()

2.20 *User Help*

File Name: UserHelp.cls

Description: This object contains the attributes associated with the help window.

2.20.1 *Properties*

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
hwnd	Long	Private		

2.20.2 *Public Methods*

Sub HTMLHelpContents()

Displays HTML help with *Contents* selected.

Sub HTMLHelpIndex()

Displays HTML help with *Index* selected.

Sub HTMLHelpSearch ()

Displays HTML help with *Search* selected.

2.20.3 *Private Methods*

Function HTMLHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" () As Long

Windows API function to display HTML help with *Contents* selected.

Function HTMLHelpTopic Lib "hhctrl.ocx" Alias "HtmlHelpA" () As Long

Windows API function to display HTML help with *Index* selected.

Function HTMLHelpQuery Lib "hhctrl.ocx" Alias "HtmlHelpA" () As Long

Windows API function to display HTML help with *Search* selected.

2.21 *Splash Screen*

File Name: frmSplash.frm

Description: This object contains the splash screen that appears at startup.

2.21.1 *Properties*

None.

2.21.2 *Public Methods*

None.

2.21.3 *Private Methods*

None.

2.22 *Method Choice*

File Name: frmMethodChoice.frm

Description: This object allows the user to choose the sample method for a particular sample parameter.

2.22.1 *Properties*

None.

2.22.2 *Public Methods*

None.

2.22.3 *Private Methods*

None.

2.22.4 *Private Events Used*

Sub cboMethods_Click()

Sub cboWellCoreType_Click()

Sub cmdCancel_Click()

Sub cmdDelete_Click()

Sub cmdOK_Click()

Sub Form_Resize()

Sub txtX_Change()

Sub txtY_Change()

2.23 *Display Options*

File Name: frmDisplayOptions.frm

Description: This object allows the user to select what items to display (or not to display) on the site.

2.23.1 *Properties*

None.

2.23.2 *Public Methods*

None.

2.23.3 *Private Methods*

None.

2.23.4 *Private Events Used*

Sub cmdApply_Click()

Sub cmdCancel_Click()

Sub cmdOK_Click()

2.24 *About Dialog*

File Name: frmAbout.frm

Description: This object shows the user an *About* window on Sim-Site. This file was provided by Visual Basic as one of their templates; therefore, it will not be described here.

2.25 File

File Name: File.cls

Description: This object represents a data file.

2.25.1 Properties

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
CurrentField	String	X		
Filename	String	X	X	
FileNum	Integer	X		

2.25.2 Public Methods

Sub CloseFile()

Closes the file.

Sub OpenFile()

Opens the file and assigns a handle.

Function GetFileName() As String

Gets a file name from the file dialog.

Function GetNextField() As String

Gets the next field from an input file.

Function GetNextLine() As String

Gets the next line from an input file.

2.25.3 Private Methods

None.

2.26 Printer

File Name: Printer.cls

Description: This object represents the current print option settings.

2.26.1 Properties

<u>Property</u>	<u>Data Type</u>	<u>Get</u>	<u>Let</u>	<u>Set</u>
BeginPage	Integer	X	X	
EndPage	Integer	X	X	
NumCopies	Integer	X	X	
Orientation	Integer	X	X	

2.26.2 Public Methods

Sub SetupPrinter()

Sets the printer options without opening the printer dialog.

Sub ShowPrinter()

Gets the printer options from the user using the printer dialog.

2.26.3 Private Methods

None.

Appendix B: Coding Conventions

1. Introduction

Coding conventions are programming guidelines that focus not on the logic of the program but on its physical structure and appearance. The main reason for using a consistent set of coding conventions is to standardize the structure and coding style of an application so that the code is easier to read, understand, and maintain. Coding conventions can include:

- Naming conventions for objects, variables, and procedures.
- Standardized formats for labeling and commenting code.
- Guidelines for spacing, formatting, and indenting.

The following sections present a set of Visual Basic coding conventions used for developing Sim-Site.

2. Coding Conventions Used for Sim-Site

2.1 Object Naming Conventions

All form and control objects intrinsic to VB6 are identified by a generic, lowercase, three-letter prefix specifying the class of the object. All form and control object prefixes used in Sim-Site are listed in Table 4.

Table 4 Prefixes for Visual Basic's Form and Intrinsic Control Objects

Class/Object Name	Prefix	Example
CheckBox	chk	chkWells
ComboBox	cbo	cboWellCoreType
CommandButton	cmd	cmdCancel
CommonDialog	dlg	dlgFileDialog
Form	frm	frmSimulator
Frame	fra	fraFeatureObjects
HScrollBar	hsb	hsbSiteImage
ImageList	ils	ilsToolBarButtons
ImgEdit	img	imgSiteEdit

Class/Object Name	Prefix	Example
Label	lbl	lblWellCoreType
Line	lin	linLine
Menu	mnu	mnuSampleResults
MS Chart	ch	chSeasonResults
PictureBox	pic	picSiteMoving
PictureClip	clp	clpSiteClip
RichTextBox	rtf	rtfResults
Shape	shp	shpGridFrame
StatusBar	sta	staCoordinates
TextBox	txt	txtEstBlock
ToolBar	tlb	tlbSimSite
VScrollBar	vsb	vsbSiteImage

2.2 Constant and Variable Naming Conventions

Each variable in Sim-Site is explicitly declared. To force explicit declaration, *Option Explicit* is included at the beginning of every code module. Declaring all variables reduces the number of bugs caused by typos. Additionally, all variable declarations have the data type explicitly stated using the *As* keyword to ensure all variables have a specific data type. Examples are as follows:

- Procedure-level scope:

```
Dim Title As String
Static KeyNum As Long
```

- Module-level scope:

```
Private mblnInitialized As Boolean
Private Const mbytLineFeed As Byte = 10
```

The scope of Visual Basic variables is shown in Table 5. The scope of variables use in Sim-Site is specified in the following way:

- Any variable declared to have module-level scope has a prefix *m* followed by a three-letter prefix to indicate a variable's data type. The data type prefixes used in Sim-Site are listed in Table 6.
- All variables of module-level scope are declared with the *Private* statement. Procedure-level variables, declared with the *Dim* statement, are private by definition. This practice conforms to the encapsulation characteristic of object-oriented programming, which forbids the use of any variables declared with the *Public* statement. Consequently, no variables with global scope are used in Sim-Site.

Table 5 Scope of Visual Basic Variables

Scope	Declaration	Visible in
Procedure-level	'Private' in procedure, sub, or function	The procedure in which it is declared
Module-level	'Private' in the declarations section of a form, code module, or class (.frm, .bas, .cls)	Every procedure in the form, code module, or class
Global	'Public' in the declarations section of a code module (.bas)	Everywhere in the application

Table 6 Variable Data Type Prefixes

Data type	Prefix	Example
Boolean	bln	mblnInitialized
Byte	byt	mbytLineFeed
Form	frm	mfrmSampleResults
Collection object	col	mcolDisplayText
Double	dbl	mdblClipPointXMin
Integer	int	mintPictureIndex
Object	obj	mobjScenario
Single	sng	msngXPosition
String	str	mstrFilename

If a Visual Basic intrinsic constant exists, it is used as shown in the following

example:

```
Msg = "Current Characterization is not saved" & Chr$(10) & _  
      "Do you want to save it before proceeding?"  
CheckForSave = MsgBox(Prompt:=Msg, _  
                      Buttons:=vbYesNoCancel + vbQuestion, _  
                      Title:="Save before proceeding?")  
If CheckForSave = vbCancel Then  
    Exit Function  
ElseIf CheckForSave = vbYes Then  
    mnuSaveAs_Click  
End If
```

If no Visual Basic intrinsic constants exist, the code declares its own constants as

shown in the following example:

```
Private Const mbytSpace As Byte = 32  
Private Const mbytLineFeed As Byte = 10  
Private Const mbytCarriageReturn As Byte = 13  
Private Const mbytQuote As Byte = 34
```

The body of a variable or procedure name is mixed case and made as long as necessary to describe its purpose. In addition, most function names begin with a verb, such as *OpenFile* or *Add*, with a few exceptions. Abbreviations may be used to help keep name lengths reasonable.

2.3 *Structured Coding Conventions*

All procedures and functions begin with a brief comment describing the functional characteristics of the procedure (what it does). This description does not include implementation details (how it does it) because these often change over time, resulting in unnecessary comment maintenance work, or worse yet, erroneous comments. The code itself and any necessary inline comments will describe the implementation.

Arguments passed to a procedure are described when their functions are not obvious and when the procedure expects the arguments to be in a specific range.

Function return values and variables that are changed by the procedure, especially through reference arguments, are described at the beginning of each procedure.

Procedure header comment blocks should include the information listed in Table 7 as appropriate. An example of a procedure header is as follows:

```

-----
' Purpose: Adds a DisplayText object to a collection of
'           DisplayText objects and returns a key (index) for the
'           object.
' Inputs:
'   All of the property values from DisplayText class.
' Returns:
'   The key to reference the object in the collection by.
-----

```

Table 7 Information for Procedure Header Comment Blocks

Section heading	Comment description
Purpose	What the procedure does (not how).
Assumptions	List of each external variable, control, open file, or other element that is not obvious.
Effects	List of each affected external variable, control, or file and the effect it has (only if this is not obvious).
Inputs	Each argument that may not be obvious. Arguments are on a separate line with inline comments.
Returns	Explanation of the values returned by functions.

Comment lines consist of an apostrophe followed by a space and then the comment. All comments appear above the lines of code to which they are related:

```

' Initialize the object
mblnInitialized = True

```

No in-line comments are used except after variable declarations:

```

Private mblnInitialized As Boolean ' = True if Simulator was
                                   ' initialized properly

```

Control-flow code blocks are indented three spaces from the enclosing code and are most often preceded by a blank line as shown:

```

Private Sub mnuOpen_Click()
    Dim Filename As String ' The user filename selected by the

```

```

' user

' Check if save is needed
If CheckForSave = vbCancel Then
    Exit Sub
End If

' Get the filename from the user
Filename = GetFileName(Filter:="User Files (*.usr)|*.usr|" & _
                        "All files (*.*)|*.*|", _
                        Title:="Open User File")

If Filename <> "" Then
    ' User did not select Cancel
    mstrUserFilename = Me.dlgFileDialog.Filename

    ' Open the user file
    Me.OpenUserFile

    ' Redraw the main window
    Me.SetFocus
    Me.Refresh
    Me.tlbSimSite.Refresh
End If
End Sub

```

Visual Basic supports the use of a line-continuation character, a space followed by an underscore (_). Code statements too long for one line are continued on the next line using the line continuation character. Each continuation line is to line up logically with the code in the line above it. An example is:

```

y1 = ((Me.picSiteMoving.ScaleHeight - _
      (theLine.PTbegy - Me.picSiteMoving.ScaleTop)) + _
      Me.picSiteMoving.ScaleTop)

```

Breaking a line in the middle of a quoted string has an ampersand (&) preceding the underscore as shown in the following example:

```

Msg = "You have $" & Format(Expression:=Budget, _
                          Format:="#0.00") & _
      " left in your budget." & Chr$(10) & _
      "Are you sure you want to start the next season?"

```

Visual Basic allows procedure calls with named parameters. Named parameters permit parameter passing in any order by explicitly naming them and assigning them values. This makes the code easy to understand. This method was used throughout Sim-

Site. When a call is made to a procedure using named arguments, the line-continuation character is often used to set off each named argument. Each continuation line contains another named argument and is indented to line up with the previous named argument.

An example is:

```
theKey = mcolWells.Add( _  
    ID:=.ID, _  
    XPosition:=.XPosition, _  
    YPosition:=.YPosition, _  
    ZPosition:=.ZPosition, _  
    StatusBarText:=.StatusBarText, _  
    SPECHHEADconstflag:=.SPECHHEADconstflag, _  
    SPECHHEADvalue:=.SPECHHEADvalue, _  
    WELLFLUXconstflag:=.WELLFLUXconstflag, _  
    WELLFLUXvalue:=.WELLFLUXvalue, _  
    WELLCONCconstflag:=.WELLCONCconstflag, _  
    WELLCONCvalue:=.WELLCONCvalue)
```

Visual Basic has two ways of passing parameters: "by value" and "by reference."

"By value" passes the actual value to the parameter. The called procedure stores the value as a local copy of the variable; therefore, it cannot change the variable owned by the calling procedure. On the other hand, "by reference" passes the address of the variable. Therefore, if the called procedure changes the value of the parameter, it is also changes in the calling procedure. Visual Basic provides the *ByVal* keyword to declare "by value" parameters, and the *ByRef* keyword to declare "by reference" parameters. One problem with this is that these keywords are optional and the default mode is "by reference." This type parameter is seldom necessary, and its side effects are difficult to detect (Swartzfager et al., 1999). Therefore, all arguments in procedure declarations have "by value" and "by reference" explicitly defined. However, arrays can only be passed by reference.

Appendix C: Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
AFIT	Air Force Institute of Technology
ASCII	American Standard Code for Information Interchange
AWD	At Work Document
BLUE	Best Linear Unbiased Estimate
BLUP	Best Linear Unbiased Spatial Predictor
BMP	Bitmap
COM	Component Object Model
DCX	File format for multipage ZSoft PaintBrush images
GIF	Graphic Interchange Format
GIS	Geographical Information System
GMS	Groundwater Modeling System
GSLIB	Geostatistical Software Library
GUI	Graphical User Interface
IDA	Inverse Distance Squared
IWDA	Inverse Distance Weighted Averaging
JPEG	Joint Photographics Experts Group
JPG	Joint Photographics Experts Group (JPEG)
MAE	Mean Absolute Error
MODFLOW	Modular Finite-Difference Ground-Water Flow Model
MSE	Mean Squared Error
ODA	Optimal Inverse Distance
OLE	Object Linking and Embedding

PCX	File format for single-page ZSoft PaintBrush images
RPM	Remedial Project Manager
Sim-Site	Site Characterization Simulator
TIFF	Tag(ged) Image File Format
TSA	Trend Surface Analysis
VB	Visual Basic
VB6	Visual Basic 6.0
XIF	File format for Xerox images

Appendix D: Definitions

ActiveX - All component technologies, other than OLE, that are built on the Microsoft Component Object Model (COM).

Ancillary Information - in remote sensing, the extra information for an area of interest used to enhance the analysis of the primary remotely sensed data. Ancillary information frequently includes topography, land cover or climate data.

Anisotropic - a descriptor for a physical property (e.g. density, etc.) that varies depending on the direction in which it is measured.

ASCII - a code in which the numbers from 0 to 255 stand for letters, numbers, punctuation marks, and other characters. ASCII code is standardized to facilitate transmitting text between computers or between a computer and a peripheral device.

Autocorrelation - a term referring to the degree of relationship that exists between two or more spatial variables, such that when one changes, the other(s) also change. This change can either be in the same direction, which is a positive autocorrelation, or in the opposite direction, which is a negative autocorrelation. For example, soil type and vegetation may be highly correlated, either positively or negatively depending upon the type of soil and vegetation under examination. Popular measures of spatial autocorrelation with spatial analysis are Moran's I and Geary's C.

AWD - a file format that supports black-and-white images only. AWD files are compressed using RBA compression.

Best Linear Unbiased Estimate (BLUE) - the result of an approximate interpolation method that recognizes that the variation of any spatial feature, such as soil properties, may be too irregular to be modeled by a smooth mathematical surface, but can be represented more appropriately using a stochastic surface.

Block Kriging - a variation of the interpolation technique kriging, one that estimates average values over an area, or block, as opposed to exact values as is the case with simple kriging.

BMP - a file format used by the Windows Paintbrush applet. BMP (bitmap) supports single-page color images.

A bitmap defines an image as a pattern of dots (pixels). A bitmap has the file name extensions .bmp or .dib. Bitmaps are also called "paint-type" graphics. You can use bitmaps of various color depths, including 2, 4, 8, 16, 24, and 32-bits, but a bitmap only displays correctly if the display device supports the color depth used by the bitmap. For example, an 8-bit-per-pixel (256 color) bitmap only displays in 16 colors when shown on a 4-bit-per-pixel (16 color) device.

Class - a group of objects that have the same attributes, behaviors, or structure.

Cokriging - a variation of the interpolation technique kriging, one that attempts to compensate for missing values in the primary variable by utilizing secondary variable data, known as covariate data.

COM Object - an object that conforms to the OLE Component Object Model (COM).

A COM object is an instance of an object definition, which specifies the object's data and one or more implementations of interfaces on the object. Clients interact

with a COM object only through its interfaces. See also Component Object Model and Interface.

Component Object Model (COM) - An open architecture for cross-platform development of client/server applications. It is based on object-oriented technology as agreed upon by Digital Equipment Corporation and Microsoft Corporation. COM defines the interface (similar to an abstract base class), IUnknown, from which all COM-compatible classes are derived.

Cursor - a bitmap that contains a hot spot, which is a pixel that tracks the location of the cursor by its x and y coordinates. Cursors have the file name extension .cur.

DCX - a file format for multipage ZSoft PaintBrush images. DCX supports both black-and-white and color images. DCX files are compressed using PCX compression.

Dynamic Model - a description of aspects of a system concerned with control, including time, sequencing of operations, and interaction of objects.

Estimation Block - the volume of media in the actual aquifer site from which the sample is composed, and over which the parameter value, estimated by the sampling method, is averaged.

Event - (in Visual Basic) an occurrence--such as a click, a double-click, or a value change--that causes an object to respond.

First Order Effect - a term used in spatial analysis to describe the behavior of spatial phenomena. First order effects manifest as variation in the mean value of the observed process in space, suggesting a global or large-scale trend. For example, the variation in precipitation across northern England displays first order effects, as the mean amount of precipitation varies due to changes in relief.

Functional Model - description of aspects of a system that transform values using functions, mappings, constraints, and functional dependencies.

Geographical Information System (GIS) - a computer system for capturing, storing, checking, integrating, manipulating, analyzing and displaying data related to positions on the Earth's surface. Typically, a Geographical Information System (or Spatial Information System) is used for handling maps of one kind or another. These might be represented as several different layers where each layer holds data about a particular kind of feature. Each feature is linked to a position on the graphical image of a map.

Layers of data are organized to be studied and to perform statistical analysis.

Uses are primarily government related, town planning, local authority and public utility management, environmental, resource management, engineering, business, marketing, and distribution.

GIF - a compressed bitmap format originally developed by CompuServe. It supports up to 256 colors and is a popular file format on the Internet.

Icon - a special kind of bitmap with a maximum size of 32 pixels by 32 pixels, but under Microsoft Windows 95, icons are also found in 16 by 16 pixel size. An icon has the file name extension .ico.

Inheritance - a feature of object-oriented programming that endows an instance of a class with the same properties as all the other instances of the same class.

Interpolation - an estimation of Z values of a surface at an unsampled point based on the known Z values of surrounding points.

Isotropic - see **Isotropy**.

Isotropy - an isotropic process, or a process that displays isotropy is one which not only exhibits **stationarity**, but where the covariance of the data depends only upon distance between the objects, and not direction.

JPEG - a compressed bitmap format, which supports 8- and 24-bit color. It is a popular file format on the Internet.

Kriging - a stochastic interpolation technique based upon a generalized least-squares algorithm, using variograms as weighting functions. Kriging is often used in obtaining estimates of surface elevation using known elevation at specific points, however, the technique can be applied to any phenomenon where a surface needs to be created from point data.

The technique is named after the South African mining geologist D. G. Krige, who developed a preliminary version of the method.

Today, a number of variants of kriging are in general use, these are: simple kriging, ordinary kriging, universal kriging, block kriging, co-kriging and disjunctive kriging.

Metafile - an image defined by coded lines and shapes. Conventional metafiles have the file name extension .wmf. Enhanced metafiles have the file name extension .emf. Only files that are compatible with Microsoft Windows can be loaded. Metafiles are also called "draw-type" graphics.

Method - a behavior built into a control. A method is a procedure that a control knows how to follow automatically.

Nonstationarity - a term used in spatial analysis to describe the presence of **first order effects** in the data. This means that the data are not mutually independent of

absolute location, as they display heterogeneity in the mean value. This suggests that although clustering may be present, this may be due to local attracting forces rather than mutual attraction of the sample data.

Nugget Effect - used to characterize the residual influence of all variabilities which have ranges much smaller than the available distances of observations. A nugget effect will appear on the variogram (or covariance) as an apparent discontinuity at the origin.

Object - an instance of a class. An object has properties, **events** and methods.

Object Model - a description of the structure of objects in a system including their identity, relationships to other objects, attributes, and operations.

OLE - an object-based technology for sharing information and services across process and machine boundaries.

Ordinary Kriging - a variation of the interpolation technique kriging, one that implicitly estimates the first order component of the data and compensates for this accordingly. This technique enables interpolation without the necessity of explicitly knowing the first order component of the data a priori.

PCX - a file format for single-page ZSoft PaintBrush images. PCX supports both black-and-white and color images. PCX files are compressed using PCX compression.

Polymorphism - ability to redefine a routine in a derived class (that is, a class that inherited its data structures and routines from another class). Polymorphism allows the programmer to define a base class that includes routines that perform standard operations on groups of related objects, without regard to the exact type

of each object. The programmer then reduces the routines in the derived class for each type, taking into account the characteristics of the object.

Property - an attribute of an object. You can set values to determine the characteristics of the object or aspects of the object's behavior.

Second Order Effect - a term used in spatial analysis to describe the behavior of spatial phenomena. Second order effects result from covariance within the data, that is, two or more variables are dependent upon one another in space. For example, the price of a house is dependent not only upon its individual characteristics, but also upon the prices of other houses within its immediate vicinity. These effects normally manifest themselves as local or small-scale pattern.

Semivariogram - see **Variogram**.

Simple Kriging - a variation of the interpolation technique kriging, one where information relating to the data are known a priori. For example, in simple kriging, the first order component of the data is known, and can be subtracted from the original sample to provide a set of residuals. This simplifies the problem of interpolation, although it is often less applicable than other forms of kriging because of the assumptions that this particular technique requires.

Smoothing - a technique used to remove or reduce local noise or high frequency signal within **spatial data**, and therefore reveal the global pattern or trend. Smoothing is a technique used in both spatial analysis and digital image processing using a variety of methods, but normally based upon a matrix or filter that passes over the image. Certain interpolation techniques, particularly the approximate interpolation methods, are also used for smoothing images.

Spatial Analysis - an analytical technique associated with the study of locations of geographic phenomena together with their spatial dimensions and their associated attributes. Spatial analysis is useful for evaluating suitability, for estimating and predicting, and for interpreting and understanding the location and distribution of geographic features and phenomena.

Spatial Autocorrelation - see **Autocorrelation**.

Spatial Data - any information about the location and shape of, and relationships among, geographic features. This includes remotely sensed data as well as map data.

Spatial Information - information which includes a reference to a two or three-dimensional position in space as one of its attributes.

Spline - a spline is a mathematical curve that often takes the form of a polynomial curve or surface that is used for purposes of **smoothing** or interpolation. Splines are essentially mathematical equivalents to the French artists curve which is a flexible plastic ruler for drawing smooth waving lines. In interpolation, a spline is often classified as an approximate, deterministic interpolation technique.

Spline Smoothing - see **Splining**.

Splining - a method of smoothing that uses splines. Splining is considered as an approximate, deterministic interpolation technique as it is based upon a mathematical function that is normally represented as a polynomial.

Stationarity - a term used in spatial analysis to describe independence within the data. A spatial process is said to be stationary or homogeneous if its statistical properties are independent of absolute location. This means that the mean and variance are constant over space and therefore do not depend upon location. Stationarity in the

second order component of spatial data is a useful, and often necessary assumption to make, as many spatial statistical methods rest upon stationarity.

TIFF - a file format for that supports multipage black-and-white, gray-scale, and color images; several compression types; and a variety of options. TIFF is compatible with all image densities.

Universal Kriging - a variation of the interpolation technique kriging, which attempts to compensate for the presence of local trends or drift in the data.

Variogram - a measure of the continuity of spatial phenomena expressed as an average squared difference between measured quantities at different locations.

Variograms, more commonly referred to as semi-variograms, are used in the interpolation technique known as kriging.

Weighting - a scaling factor that expresses the importance of a variable in a particular operation. The weighting will determine the importance or influence that a particular object or element will have on the final result of the operation.

XIF - a file format for Xerox images. XIF supports multipage black-and-white images only. XIF files can be uncompressed, or compressed using Xerox-proprietary compression.

Bibliography

- Adobe Developers Association. *TIFF 6.0 Specification*. Adobe Systems Incorporated, Mountain View, California, June 3, 1992.
- Birse, C. and Marinkovich, M. *Understanding PackBits 2, Technote 1023 - Release 1.0*. Apple Developer Technical Support (DTS), Apple Computer, Inc. January 22, 1996.
- Engineering Computer Graphics Laboratory. *GMS v2.1 File Formats*. Brigham Young University, Provo, Utah, February 14, 1998.
- Engineering Computer Graphics Laboratory. *GMS v2.1 Reference Manual*. Brigham Young University, Provo, Utah, February 17, 1998.
- Collins, F. and Bolstad, P. *A Comparison of Spatial Interpolation Techniques in Temperature Estimation*. Third International Conference/Workshop on Integrating GIS and Environmental Modeling, Santa Fe, New Mexico, January 21-25, 1996.
- Deutsch, C. and Journel, A. 1992. *GSLIB: Geostatistical Software Library and User's Guide*. Oxford University Press, Inc., New York, NY. (ISBN: 0-19-507392-4)
- Harbaugh, A. and McDonald, M. *User's Documentation for MODFLOW-96, an update to the U.S. Geological Survey Modular Finite-Difference Ground-Water Flow Model, Open-File Report 96-485*. U.S. Geological Survey, Reston, Virginia, 1996.
- Heiderscheidt, J. L. *Development of Site Characterization Simulator Specifications*. MS thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH AFIT/GEE/ENV/96D-07, November 1996.
- Jimenez, A., *Special Topics in Soil, Crop & Atmospheric Sciences: Land evaluation, with emphasis on computer applications*. Department of Soil, Crop, & Atmospheric Sciences, College of Agriculture & Life Sciences, Cornell University, Ithaca, NY, 1995.
- Journel, A. and Huijbregts, C. 1978. *Mining Geostatistics*. Academic Press, Inc., San Diego, CA. (ISBN: 0-12-391056-0)
- Microsoft Corporation, 5 August 1998. *Microsoft Visual Basic 6.0 Programmer's Guide*. (ISBN: 1-57231-863-5)
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. 1991. *Object-Oriented Modeling and Design*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey. (ISBN: 0-13-629841-9).

Swartzfager, G., Chandak, R., Chandak, P., and Alvarez, S. 1999. *Visual Basic 6 Object-Oriented Programming Gold Book*. The Coriolis Group, Inc., Scottsdale, Arizona. (ISBN: 1-57610-255-6)